(2)

高精度CAD/CAMのための形状モデリングに関する研究

渡部広一

まえがき

コンピュータ支援による設計／製造、すなわち、CAD/CAM はコンピュータの計算・情報処理能力を適用して、人間の知的活動である設計・製造のあらゆる場面で、その効率化と質的改善を目指す理論・技術の体系および応用分野を指す。中でも、製品の形状をコンピュータ内に表現するための形状モデルの構築法、および、その形状モデルを利用して、設計・製造のためのあらゆる情報を取り出すためのシミュレーションプログラムの開発は、形状モデリングの問題として、CAD/CAM の中で中心的役割を果たしている。

本論文は、高精度な CAD/CAM システムを構成するための高精度で高速な形状モデリング手法の開発を行い、形状モデルの高精度・高機能化を図り、さらにいくつかのシミュレーションプログラムへの適用を行った研究結果をまとめたものである。すなわち、設計・製造のための形状情報をソリッドモデルとして計算機内に格納するためのモデリング手法の開発と、その格納されたソリッドモデルから、目的の製品に関する情報、および、製品の設計・製造過程のシミュレーションに必要となる情報を抽出するための形状モデリング手法の開発である。

そのために、まず、形状モデリングのための中心的な道具となる相貫曲線を高精度・高速に求める手法を提案する。ここで扱う曲面は、平面、2 次曲面およびトーラス（4 次曲面）で、それらの曲面間の相貫曲線を解析的に求める。すなわち、相貫曲線の式を与える。

次に、ソリッド形状モデルの高精度化のために、CSG 表現において困難とされている曲面間の接合部分にフィレットを発生させる問題を扱う。また、ソリッド形状モデルの高機能化のために、ソリッドモデルの 2 つの代表的な表現法であるところの、CSG（Constructive Solid Geometry）から B-Reps（Boundary Representations）への変換を近似をせずに高速に精度良く行う手法を開発し、その結果として CSG/B-Reps の二重構造モデルを構築する。CSG から B-Reps への変換によって、それぞれの表現方式の短所をおぎない相手の長所を取り入れたソリッドモデル表現が可能となる。このような変換の高精度性、高速性の達成のために、相貫曲線の解析解の利用が有効であることを示す。

また、シミュレーションプログラムへの適用として、製品形状モデルから金型形状モデルを自動生成する問題、および、形状モデル間の干渉認識手法の開発を通して、製品の組立順序を自動生成する問題を扱う。干渉認識は単なる干渉チェックとは違い、干渉のあるなしの判定はもちろん形状モデル間の接触状態の認識を行う点に特徴がある。

これらのシミュレーションプログラムは、製品モデルとしてCSG/B-Reps二重構造モデルを利用している。

最後に、形状モデルをベースとしてCAD/CAMシステムを構築するときの環境そのものに対して新しい提案をする。すなわち、各種の形状モデリング手法をプログラム化し、さらに、目的に応じてその他各種のプログラムを作成し、それらを結合することによってCAD/CAMシステムを構築するわけであるが、その作業は大変なものであり、また、いったんできたものを修正・管理することも非常な困難が付きまとう。このような現状を少しでも打開するために、自律駆動型プログラムモジュール（ADPM）という概念を導入する。そして、ADPMを利用してCAD/CAMシステム構築用部品を作成し、その集合として目的にあったCAD/CAMシステムを構成できるような環境を提供することを目標とし、その基礎研究を行った。各プログラムをADPMとすることによって、非常に自由度の高いCAD/CAMシステムを構成できる可能性を示す。

目次

図 目 次

第 1 章

序論；本研究の背景と目的

1.1 CAD/CAM の構成

CAD/CAM とは何かを一言で述べると、「コンピュータを利用することにより生産活動の効率化と質的改善を図ること」と言える。生産活動とは、何らかの「もの」を作ることであり、その中には、人の心の中にある漠然とした「もの」のイメージを具体的な他の人に分かる情報に変換する作業、すなわち、設計と、その設計情報から、実際の「もの」を作る過程、すなわち、製造とが含まれる。これらをコンピュータの高度な計算・情報処理能力を利用して行うとき、前者を CAD と呼び、後者を CAM と呼ぶ。図 1.1 はこのような CAD/CAM の構成を示す。以下、この図に沿って CAD/CAM の概念を説明する。

まず、設計者は製品の仕様が与えられると、その製品の形状や構造のイメージを頭に思い描きながら、構造設計・部品設計を行い、その結果として製品モデルを作成する。製品モデルの中には、製品形状モデル、製品各部の属性、設計諸元などが含まれる [2]。一旦製品モデルが出来上がったならば、それに対して各種の性能解析（シミュレーション）を行い、その製品モデルが与えられた仕様を満足するかどうかを調べる。これには、製品モデルの解析（グラフィック表示、マスプロパティ計算、強度・熱・振動解析など）、および、製造のシミュレーション（加工法の検討、組立可能性の判定、ロボットによる作業のシミュレーションなど）が含まれる。このようなシミュレーションの結果がおもわしくなければ、設計者は構造設計、部品設計などをやり直して、製品モデルの修正を行う。そして、このような製品モデルの作成・修正とシミュレーションによる製品モデルの評価を繰り返すことにより満足のできる製品モデルを完成させる。製品モデルが完成した後は、製造のシミュレーションを実際の機械に置き替えて実行することによって製品が完成する。

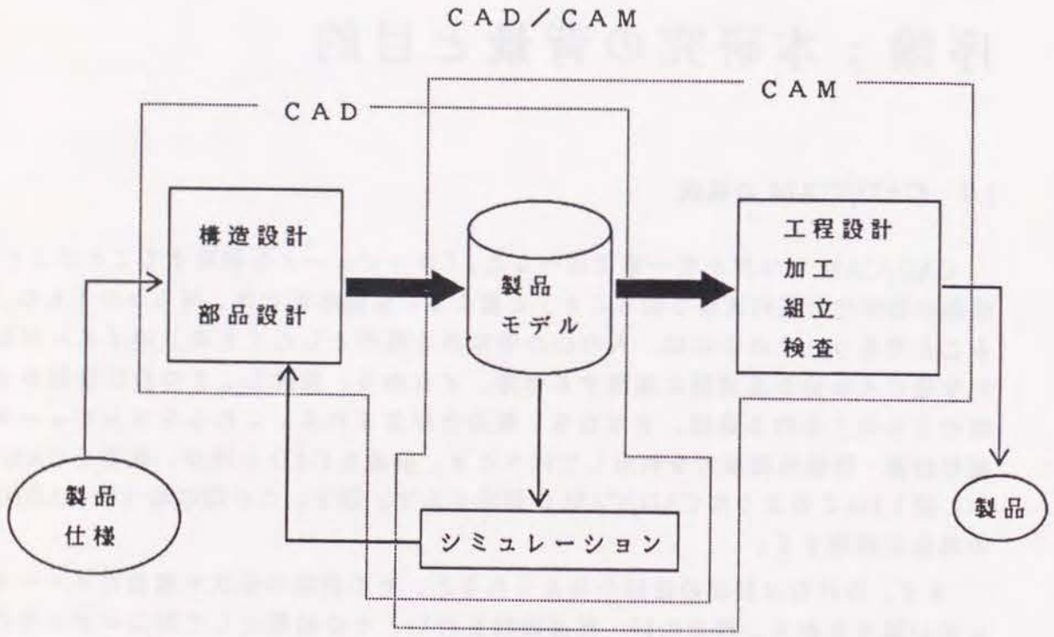


図 1.1: CAD/CAM の構成

製品モデルの中で、製品各部の属性は製品形状と関連づけられて決定されるものであり、設計諸元も製品形状と無関係に決定できるものは少ない。したがって、製品の形状モデルが基本的な役割を果たすと考えられる。また、形状モデルとシミュレーションは互いに密接な関係にある。すなわち、形状モデルに内在的に含まれていないことはシミュレーションできないし、内在的には含まれていることでも表現の仕方によっては、その情報を取り出せないためにシミュレーションプログラムの作成が困難になる。

そこで本研究では、CAD/CAM において特に重要な形状モデリングの問題を中心に扱う。すなわち、3 次元的な形状をどのように計算機内に構築し、その 3 次元形状モデルからいかにして効率よく信頼性の高いシミュレーションを行うかという問題である。そこには、形状モデルの高精度性・高機能性が要求され、その実現によってシミュレーションの高信頼性が達成される。

1.2 形状モデリング

形状モデリングには、計算機内にいかに形状を表現するかの問題、すなわち、形状モデルの構築と、その形状モデルからどのように設計・製造のための情報を取り出すかの問題、すなわち、シミュレーションプログラム側での形状処理に関する部分が含まれる。狭い意味では前者のみを指して形状モデリングと呼ぶこともあるが、本論文では後者も含める。

したがって、本研究では形状モデリングの問題として、CAD/CAM における製品形状モデルの構築と、その製品形状モデルを利用して設計・製造のためのシミュレーションプログラムを作成する問題を扱う。この際、効率よく信頼性の高いシミュレーションを行うためには、高精度で高機能な形状モデルの構築が必要となる。

1.3 形状モデル

形状モデルは、設計者が何らかの形で計算機に入力するものであるから、入力の容易さが求められるが、それだけではなく、形状表現能力が十分高く、さらに、各種のシミュレーションに効率よく利用できなければならない。そこには、

1. 形状表現能力を高めると入力が難しくなる。

2. 形状表現能力が低いと入力がやさしい代わりに必要なシミュレーションが原理的に不可能、あるいは、信頼性が低くなる。

3. 形状表現能力が十分高くても、複雑になりすぎたためにシミュレーションプログラムの作成が困難、あるいは、作成できるとしても処理時間がかかりすぎて実用的でなくなる。

などの問題が発生する。したがって、形状モデリングの問題として、上のように互いに相反する要件を満足することが要求されている。

3 次元形状モデルとしては、形状表現能力の低い順に、ワイヤーモデル、サーフェイスモデル、ソリッドモデルがある。ワイヤモデルは線のみによって形状を表現するので形状表現能力が低く簡単なことしかできない。サーフェイスモデルは形状を面の集合で表すもので、使用目的によってはこれで十分なこともあるが、3 次元形状を完全に表しているわけではないので問題が残る。ソリッドモデル [4] は 3 次元形状を完全に表現することが可能になる。しかし反面、形状表現能力が高いがゆえに、上の 1 、3 の問題が発生することになる。形状表現能力が基本的に低いモデルを使用することは、入力の容易さ、シミュレーションプログラムの作成のしやすさ、処理時間の速さなどのメリットがあるとはいえ、必要なシミュレーションができなかったり、信頼性がなかったりする結果となるので無意味である。そこで本研究は、形状モデルとして形状表現能力が十分に高いソリッドモデルを採用し、その問題点、すなわち、入力の困難さや高効率で信頼性の高いシミュレーションプログラム作成の困難さを打開するという方針で進める。

現在までにソリッドモデルの表現形式として種々のものが考えられている [5] 。中でも代表的なのは CSG (Constructive Solid Geometry) と B-Reps (Boundary Representations) である。以下、CSG と B-Reps について述べる。

1.3.1 CSG

CSG は、1973 年ハンガリーのブタペストで開催された国際会議 PROLAMAT で北海道大学のグループによって発表された TIPS-1 システム [1] で提案されたソリッドモデルの表現形式である。CSG では、3 次元形状をプリミティブと呼ばれる基本形状の集合演算（和・差・積集合）によって表現する。式で表すと 3 次元形状 S は、

$$S = (\bigcup_{i=1}^{m_P} P_i) \cup (\bigcup_{j=1}^{m_Q} Q_j) \tag{1.1}$$

となる。ただし、P_i、Q_jはそれぞれ正、負のプリミティブとする。このように、形状 S を定義するのにプリミティブを足したり引いたりするだけで、プリミティブ同志の交わり（相貫曲線など）を定義する必要はなく、また、他のプリミティブに埋没させて消す部分は、はみ出さない限り長くても短くても良いなど、設計者にとっては容易な

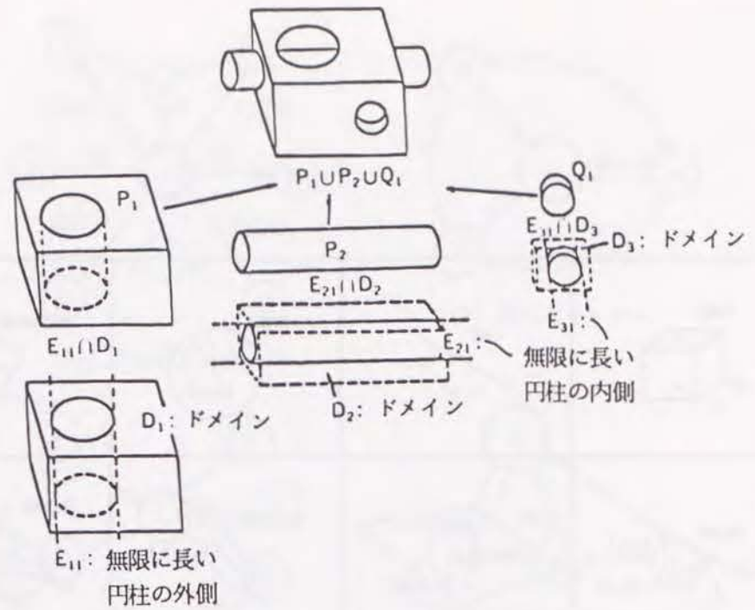


図 1.2: TIPS-1 の CSG データ構造

形状定義が可能になる。例として、TIPS-1[1] の CSG 表現を図 1.2に、主なプリミティブの種類を図 1.3に示す。

図 1.3からわかるように、CSG プリミティブは平面、2 次曲面、および、トーラス（4 次曲面）を表面として持つものがほとんどである。工業製品は、一見複雑そうに見える物でも部分的には比較的単純な形状をしており、平面と円柱面で約 6 0 パーセントの部品が記述可能と言われている [2] 。それに円錐、球などの 2 次曲面を加えることにより、さらに記述力が上がる。また、円弧に沿う部分に丸み（フィレット）を付けたい場合や、パイプのエルボー、コーヒーカップの取っ手（近似が必要）などは、トーラスで表現することができる。このように、ほとんどの製品は平面、2 次曲面、トーラスで表現可能である。本論文ではこれらの曲面を基本形状曲面と呼ぶ。もちろん、すべての製品が基本形状曲面で表現できるわけではなく、自動車のボディーや、2 次曲面同志の接合部のフィレット（これは第 2 章で扱う）など基本形状曲面では扱いきれないものもある。TIPS-1 では、このような基本形状曲面では表現できないものを扱うために、CONTOR と呼ばれる任意の自由曲面を表面として持つプリミティブを定義できるようになっているが、その取扱は困難である。

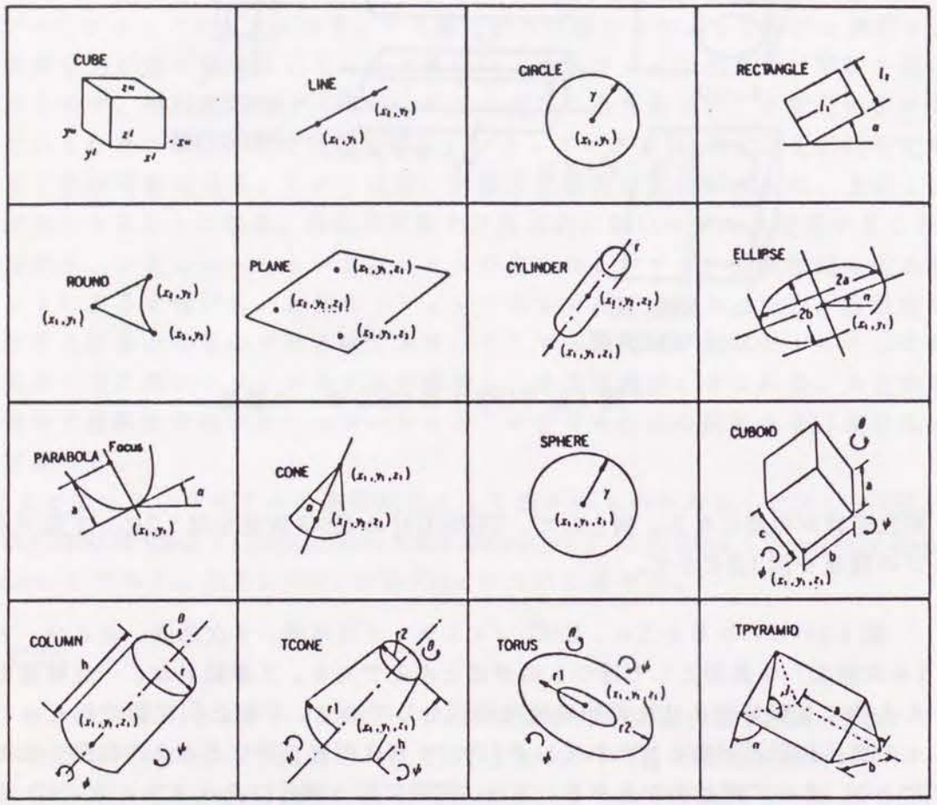
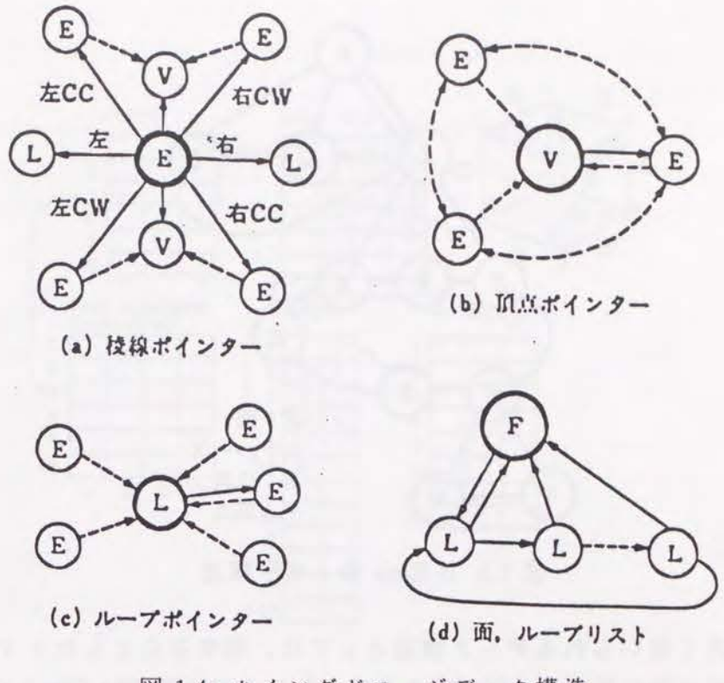


図 1.3: TIPS-1 のプリミティブ



(a) 枝線ポインター　(b) 頂点ポインター

(c) ループポインター　(d) 面，ループリスト

図 1.4: ウイングドエッジデータ構造

1.3.2 B-Reps

B-Reps も同じく、1973 年、ハンガリーのブタペストで開催された国際会議 PROLAMAT で Cambridge 大学のグループによって発表された BUILD と呼ばれるシステム [3] が翌年採用したソリッドモデルの表現形式である。B-Reps では、形状の表面を構成する面、稜線、頂点の幾何データと、それらの結合関係（トポロジーデータ）によって 3 次元形状を表現する。

結合関係を表すデータ構造としては、初め Baumgart[6] が提案し、後に Braid[7] が拡張したウイングドエッジデータ構造が有名である（図 1.4）。この構造の大きな特徴は、すべての稜線について (a) の型は同じであるから、これを収容するためのメモリーサイズは常に一定で固定フォーマットをとれることである。このことは、データ処理の単純化に有効である。また、一つの稜線からループをなす隣の稜線を知りたいとき、時計回り（ cw ）、半時計回り（ cc ）のいずれでもただちに知ることができ、その稜線のウイングドエッジへ移ることを繰り返せば容易にループをピックアップできる。ただし、この特徴は冗長なデータの格納状態を許すことによって得られるものであり、多量のデータを要し、かつ処理時間も長くなる。

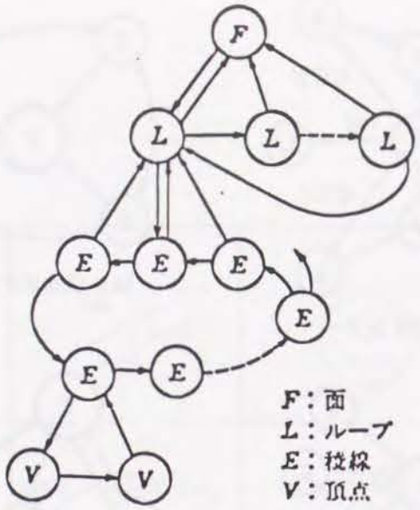


図 1.5: B-Reps のトリー構造

もう一つ良く用いられるデータ構造としては、面を中心としたトリー構造がある（図 1.5）。形状は面の集まり（面は向きを持つ）であり、面はいくつかのループからなる。ループは稜線を並べて閉じたものであり、稜線は両端に頂点を持つ。トリー構造は、この状態を忠実に表現したものとみることができる。

1.3.3 CSG と B-Reps の比較

1.3.1節で述べたように、CSG ではプリミティブと呼ばれる比較的単純な立体を組み合わせることにより（集合演算）、目的の 3 次元形状を定義する。定義データとして必要なものは、プリミティブの種類、プリミティブの形状のパラメータ（円柱の半径・長さ、直方体の各辺の長さなど）、プリミティブの位置・姿勢、演算子（和・差・積）などである。

プリミティブ自身が立体であるので、形状定義のどの時点でも全体として立体になっており、その点に関して設計者は注意を払う必要はない。しかし、定義結果として得られるものは、プリミティブの集合体だけであるので、それが目的の 3 次元形状を表しているとはいえ、シミュレーションプログラム側では、そのままでは扱いづらいことがある。例えば、形状の斜視図を描きたい場合は、各プリミティブ間の交わり部分に発生する稜線を計算しなければならないし、他のプリミティブに埋没したり差演算で除去されている部分は不用部分として取り除かなくてはいけない。

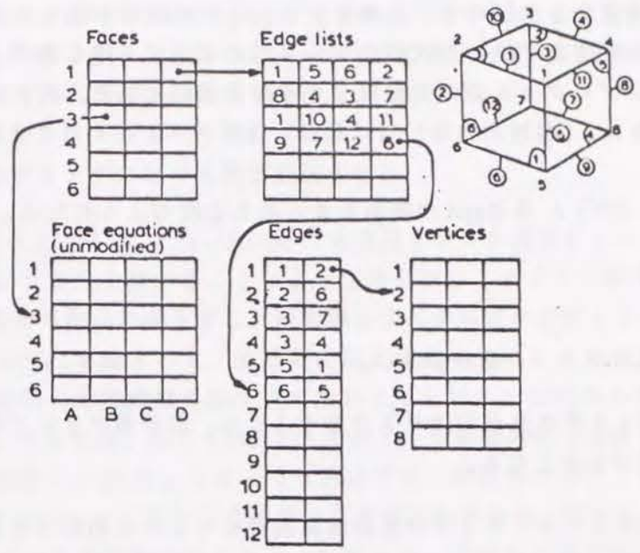


図 1.6: 直方体の B-Reps データ構造（ BUILD ）

このように CSG では、集合演算子付きのプリミティブがそのままの形で格納されているだけであるので、実際の形状処理はシミュレーションプログラム側にまかされることとなり、シミュレーションプログラムの負担が重くなる傾向にある。

それに対して B-Reps では、先に述べたように、幾何データ（面、稜線、頂点）とトポロジーデータ（幾何データ間の接続関係）によって 3 次元形状を定義する。ここで面データは曲面の式で表現され、それだけでは一般に無限に広い曲面になる。その無限領域を仕切って面分を構成するわけであるが、その仕切りがループおよび稜線である。ループは稜線をつなげて閉じたもので、幾何データはなくトポロジーデータのみである。稜線は（一般に無限に長い）線の幾何データ、すなわち、曲線式と両端点（頂点）により構成される。

一般に B-Reps では、3 次元形状を表現するための幾何要素の数は多くなり、それとともにトポロジーデータも多く複雑になる。図 1.6に示すように、単純な形状でも B-Reps で表現すると、そのデータ量が大きく複雑であることが分かるであろう。

さらに、B-Reps の構成要素は面、稜線、頂点のどれをとっても立体ではなく、立体表面の一部を成しているにすぎない。これらの要素を 3 次元的に過不足なく継ぎ合

わせて閉じた空間を作るため、どこかに不備が発生すると、もはや 3 次元形状ではなくなり、ここにも形状定義の困難さがある。また同じ理由で、形状内外判定も形状表面データから直接計算できないので、工夫を要し [10] 計算効率が悪くなる。

反面、B-Reps の定義終了後は形状表面データは過不足なく陽に表現されているのでシミュレーションプログラム側の負担は少なくなる傾向にある。例えば、形状の斜視図を描きたいときは、(隠線消去をしなければ) 稜線データをそのまま使えるので容易である。

以上の考察から CSG と B-Reps の特徴をまとめると次のようになる。

CSG

1. 形状表現のためのデータ量が少ない。
2. 同一形状に対する定義法が何通りも存在するため、設計者にとっては自由度が大きく形状定義が容易になる。
3. 形状内外判定は、プリミティブの集合演算表現をそのまま利用できるので容易である。
4. 逆に、プリミティブの集合演算表現では、形状 (特に形状表面) が陽に表現されていない。そのため、シミュレーションプログラム側で、陰に表現されている形状情報を陽に表す作業が必要となることが多い。

B-Reps

1. 形状表現のためのデータ量が多く、データ構造が複雑である。
2. そのため、これを設計者が直接入力することは困難である。
3. 形状内外判定は、形状表面情報のみからは直接計算できないので、工夫が必要となり、計算効率が悪い。
4. 形状表面情報が陽に表現されているため、シュミレーションプログラムに容易に適用できることが多い。

このように、ソリッドモデルとして代表的な CSG 、B-Reps の両表現ともそれぞれ長所短所がありどちらが良いかは一概に言えないばかりか、どちらを採用しても必ず短所を伴うことになる。そこで本研究では、基本的には CSG 表現を採用するが、必要に応じて B-Reps 表現も利用できるような CSG と B-Reps の二重構造のソリッドモデルを構築する。すなわち、形状の内外判定など形状内部情報を必要とするようなシミュレーションプログラム用には CSG を利用でき、形状の表面情報が必要な場合には B-Reps を利用できるようなソリッドモデルである。また、形状定義は入力が容易な CSG によって行い、B-Reps は自動発生させる。

1.4 形状モデリングにおける相貫曲線の抽出

前節で述べたような、CSG/Breps 二重構造モデルを構築するためには、プリミティブ同志の交わりかたを調べることによって他のプリミティブに埋没している部分を消去したり、交わり部分に発生する形状稜線を求める処理が必要となる。このような処理のための基本的な手段として、曲面間の相貫曲線を求めることが重要である。しかし従来は、曲面間の相貫曲線を精度良く求めようとすると時間がかかりすぎるため、近似的な方法、すなわち、元の CSG プリミティブを多面体化した後で相貫曲線を求める手法が採られていた [8,9] 。このような方法では、形状モデルとしての精度が落ちるため、シミュレーションにおいて支障をきたす可能性がある。例えば、形状モデルから切削用の NC データを自動生成するような場合は、形状モデルが多面体になっているため、多面体形状の製品しか削り出せない。したがって、製品形状の精度が要求されるような場合は、NC データの作成は自動的にはできず人手によらなければならなくなる。

元の CSG の形状表現精度を B-Reps においても維持するために、相貫曲線は厳密に求める必要があり、また、高速性も要求される。本研究では、そのような条件を満足するには、相貫曲線の解析的厳密解を利用するのが最適であると考えた。すなわち、相貫曲線を求めようとしている 2 つの曲面の式を並べて、これを連立方程式として解いた結果の式 (相貫曲線式) を利用する方法である。一般の曲面間の相貫曲線を解析的に求めることは困難であるが、CSG プリミティブを構成する曲面は平面、2 次曲面、トーラス (4 次曲面) がほとんどであることを考慮すると、これらの曲面間の相貫曲線の解析的厳密解を求めることで十分であると考えられる。

また、曲面間の相貫曲線は CSG から B-Reps への変換に利用できるだけではなく、CAD/CAM における形状モデリングの色々な問題で利用可能である。以下、それらの主な問題を挙げてみる。

図形処理: 形状のワイヤフレーム表示における形状稜線の抽出やスキャンライン法による面画表示において相貫曲線を求めることが必要となる。

フィレットの発生: 曲面間の接合部分にフィレットを発生させるためには、曲面間の接合部分、すなわち、相貫曲線が必要である。

NC ツールパスの生成: 加工面（製品の表面形状）を工具半径分だけオフセットした曲面と、平面あるいは曲面（加工法による）との相貫曲線がツールパスとなる。

干渉チェック： 組立製品などの場合に各部品間の干渉がないかどうかを判定する必要があるが、この干渉チェックは部品形状間の相貫曲線があるかないかで判定可能である。

金型の型分割： 製品を金型を使って製造しようとするとき、製品形状の空洞を持つ金型を作成することになる。このとき、ブランク形状（直方体）から製品形状を差し引いた形状モデルを 2 つに分割しなければならないが、そのために相貫曲線を利用する方法が有効である。

組立製品の認識： 組立製品の設計では、干渉のチェックだけでは不十分で、各部品同志がどのように接触接続しているかを認識する必要がある。この部品間の接続状態の認識には相貫曲線を求めることが重要である。

1.5 本研究の目的と方針

CAD/CAM における形状モデリングは、形状モデルの構築、および、そのシミュレーションプログラムへの適用のための基礎手法であり、CAD/CAM の中で中心的な役割を果たす。本論文の目的は、3 次元製品形状を忠実に表現可能なソリッドモデルをベースとする高精度な CAD/CAM システムを構成するための高精度で高速な形状モデリング手法を開発し、形状表現およびいくつかのシミュレーションプログラムに適用することによって、その有効性を示すことにある。すなわち、設計・製造のための形状情報をソリッドモデルとしてコンピュータ内に格納するためのモデリング手法を開発し、その格納されたソリッドモデルから、目的の製品に関する情報、および、加工、組立、金型の利用などの製品の製造過程のシミュレーションに必要となる情報を抽出するための形状モデリング手法の開発を行う。

そのために本論文では、次のような方針を取る。

1. 形状定義は入力が容易な CSG によって行う。
2. CSG の高精度化のために、CSG では取扱が困難とされている曲面間の接合部のフィレットを自動発生させる。（第 3 章）
3. CSG の短所である形状表面が陽に表現されていない点をおぎなうために、CSG から B-Reps を自動発生し、CSG/B-Reps 二重構造モデルを構築する。この際、CSG の形状表現精度を B-Reps でも保つために、多面体近似は行わず、曲面を含むプリミティブもそのまま扱う。また、CSG から B-Reps への変換に際しての高速性を実現するために、曲面間の相貫曲線の解析解を利用する。（第 4 章）
4. CSG/B-Reps 二重構造モデルの有効性を確認し、シミュレーションプログラムにおける形状モデリング手法を示すために、いくつかのシミュレーションプログラムを開発する。具体的には、金型 CAD システムへの適用（第 5 章）、干渉認識問題への適用（第 6 章）、自動組立問題への適用（第 7 章）を図る。

そして、全体を通して、形状モデリングのための基礎技法として、曲面間の相貫曲線の解析解（第 2 章）が有効であることを示す。

1.6 本論文の構成

以上のような方針の基に、本論文は以下のような構成をとる。

まず第 2 章において、形状モデリングのための中心的な道具となる相貫曲線を求める手法について述べる。ここで扱う曲面は、CSG プリミティブを構成する面のほとんどをカバーする、平面、2 次曲面およびトーラス（4 次曲面）で、それらの曲面間の相貫曲線を解析的に求める。すなわち、相貫曲線の式を与える。

第 3 章はソリッド形状モデルを高精度化するために不可欠な問題として、CSG 表現において困難とされている曲面間の接合部分にフィレットを発生させる問題を扱う。一般に、フィレットは CSG プリミティブでは表現が困難なため、その自動生成法が望まれている。曲面間の相貫曲線の解析解を利用することによって高精度なフィレットの発生が可能となることを示す。

第 4 章では、ソリッド形状モデルの高機能化のために、CSG から B-Reps への変換を近似をせずに高速に精度良く行う手法を開発し、その結果として CSG/B-Reps の二重構造モデルを構築する。すでに述べたように、CSG は基本形状のセットオペレーションによって形状を構築するのに対し、B-Reps は形状表面の幾何要素（面分、稜線、頂点）の幾何情報とそれらの接続関係によって形状を表現するものであり、それぞれ一長一短がある。CSG から B-Reps への変換によって、それぞれの表現方式の短所をおぎない相手の長所を取り入れたソリッドモデル表現が可能となる。このような変換の高精度、高速性の達成のために、相貫曲線の解析解の利用が有効であることを示す。

第 5 章は、第 4 章で構築した CSG/B-Reps の二重構造モデルを利用して、製品形状モデルから金型形状モデルを自動生成する問題を扱う。これは、CAM 用シミュレーションプログラムの一種で、金型による製造の事前検査を実際の製造の前に行うものである。

第 6 章も、第 4 章で構築した CSG/B-Reps の二重構造モデルを利用して、形状モデル間の干渉の認識を扱う。干渉認識は単なる干渉チェックとは違い、干渉のあるなしの判定はもちろん形状モデル間の接触状態の認識を行う点に特徴がある。これは、ソリッド形状モデル構築の問題と考えることもできるし、シミュレーションプログラムの問題とも見ることができる。

第 7 章は、組立のシミュレーションプログラムの一種で、組立の順序を製品モデルの情報から自動的に求める問題を扱う。製品の各部品モデルは、同じく、CSG/B-Reps の二重構造モデルを利用し、さらに、第 6 章の干渉認識手法を各部品モデル間に適用することにより、それらの幾何的接続状態を自動的に取り出し、各部品の移動可能方向などを求め、問題の解決を図る。

第 8 章は、形状モデルをベースとして CAD/CAM システムを構築するときの環境そのものに対して新しい提案をする。すなわち、前章までに開発された各手法などをプログラム化し、さらに、目的に応じて各種のプログラムを作成し、それらを結合することによって CAD/CAM システムを構築するわけであるが、その作業は大変なものであり、また、いったんできたものを修正・管理することも非常に困難が付きまとう。このような現状を少しでも打開するために、自律駆動型プログラムモジュールという概念を導入し、CAD/CAM システム構築用部品の集合として、目的にあった CAD/CAM システムを構成できるような環境を提供することを目標とし、その基礎研究を行った結果を述べる。前章までの各プログラムを自律駆動型プログラムモジュールとすることによって、非常に自由度の高い CAD/CAM システムを構成できる可能性を示す。

最後に、第 9 章で本論文の結論を述べる。

参考文献

[1] N.Okino, Y.Kakazu and H.Kubo : "TIPS-1, Technical Information Processing System for Computer Aided Design, Drawing and Munufacturing", Proc. of PROLAMAT'73, (1973)141.

[2] 沖野教郎：自動設計の方法論、養賢堂 (1982).

[3] I.C. Braid, C.A.Lang : "Computer-Aided Design of Mechanical Components with Volume Building Bricks", Proc. of PROLAMAT'73, (1973)174.

[4] A.A.G.Requicha and H.B.Voelcker : "Solid Modeling: Current Status and Research Directions", IEEE CG & A, October (1983)25.

[5] Jaroslaw R. Rossignac : "Depth-Buffering Display Techniques for Constructive Solid Geometry", IEEE CG&A, September (1986)29.

[6] B.G.Baumgart : "Geometric Modelling for Computer Vision", Stanford Artificial Intelligence Lab., report STAN-CS-74-463 (1974).

[7] I.C.Braid : "Notes on a Geometric Modeller", CAD Group Document No.101, University of Cambridge, June (1979).

[8] H. B. Voelcker et al. : "Boundary Representation & the BFILE/1 System", PADL System Document, No.10 (1977).

[9] 山口富士夫、太田健一、佐藤敬、土川仁：" 4 × 4 行列法に基づく幾何演算高速化の一手法", 精密工学会誌、55 、5 、(1989)102.

[10] Yehuda E. Kalay : "Determining the Spatial Containment of a Point in General Polyhedra", Computer Graphics and Image Processing, **19** (1982)303.

第 2 章

基本形状曲面の相貫曲線解析解

2.1 はじめに

序論で述べたように、高精度なCAD/CAMのための形状モデリングにおいて、曲面間の相貫曲線を近似をせずに高速かつ精度よく求めることが要求されている。すなわち、形状モデリングにおけるCSGからB-Repsへの変換の際のプリミティブ間の交わりを求める問題や、図形処理、フィレットの発生、NCツールパスの生成、干渉チェック、金型の型分割、組立製品の認識など、多くの形状モデリングの問題において曲面間の相貫曲線を求めることが要求される。その要求を満足するもっとも良い方法は、相貫曲線の解析的厳密解を利用することであろう。一般に解析解を求めることは困難であるが、2次曲面同志についてはいくつかの方法が存在する。一つは、2元連立2次方程式を立てて4次方程式を解く方法[1]。もう一つは、パラメータ表現を利用して2次方程式を解く問題に帰着させる方法である[2]。本研究では、2次曲面だけでなく4次曲面であるところのトーラスを含めた各形状間の相貫曲線を解析的に求めることを試みた。2次曲面と4次曲面との相貫曲線を求める問題は、代入法で解くと8次方程式を解く問題となるが、8次方程式は解析的には解けない。ところが、曲面をパラメータを使って表現すると、4次方程式（球とトーラスの場合は三角方程式）を解く問題に帰着できるので、4次方程式の解の公式を用いて相貫曲線の解析解が得られる。

本章では、トーラス、線織面（ここでは基本形状曲面として円柱、円錐）、球および平面間の相貫曲線の解析解を示す。

2.2 相貫曲線の一般的求め方

一般に、相貫曲線の解析解の求め方は次の3通り考えられる。

1. 二つの曲面の陰関数表現をそのまま連立方程式として解く方法。
2. 両曲面をパラメータ表現にして連立させる方法。
3. 一方をパラメータ表現、他方を陰関数表現として、パラメータ表現式を陰関数表現式に代入して求める方法。

以下、各々の方法について述べる。

2.2.1 陰関数法

一般に、x,y,zの関数を零で等置した式、例えば、次の二つの式はそれぞれ一つの曲面を表す。

$$f(x,y,z)=0 \tag{2.1}$$
$$g(x,y,z)=0 \tag{2.2}$$

上の式2.1,2.2をそのまま連立方程式として解く方法が陰関数法である。

この方法を、平面と平面、平面と 2 次曲面との場合に適用する。

2.2.2 パラメータ法

一般に曲面は 2 個のパラメータ s,t を使って表せる。今、二つの曲面が、関数ベクトル $\mathrm{p}=(p_1,p_2,p_3),\mathrm{q}=(q_1,q_2,q_3)$ 、パラメータ s_1,t_1,s_2,t_2を使って、次のように与えられたとする。

$$\mathbf{P} = \mathrm{p}(s_1,t_1) \tag{2.3}$$
$$\mathbf{Q} = \mathrm{q}(s_2,t_2) \tag{2.4}$$

パラメータ表現された曲面の相貫曲線は、4 個のパラメータを 3 個の関係式を使って 1 個にすれば求まることになる。すなわち、$\mathbf{P}=\mathbf{Q}$ より、以下の 3 個の式が得られる。

$$\begin{cases} p_1(s_1,t_1)=q_1(s_2,t_2) \\ p_2(s_1,t_1)=q_2(s_2,t_2) \\ p_3(s_1,t_1)=q_3(s_2,t_2) \end{cases} \tag{2.5}$$

式 2.5は未知数を s_1,t_1,s_2,t_2とする 3 元連立方程式であるから、未知数の 1 つを固定して考えることにより、他の未知数を固定した未知数の関数として求めることが可能である。例えば、

$$s_1=s_1(t_1) \tag{2.6}$$

式 2.6を式 2.3に代入すると、

$$\mathbf{P}=\mathrm{p}(s_1(t_1),t_1)=\mathrm{p}'(t_1) \tag{2.7}$$

となり、式 2.7は、曲面 $\mathbf{P}$ と曲面 $\mathbf{Q}$ との相貫曲線を表す。

トーラス同志、トーラスと球、2 次曲面同志の場合にこの方法を適用する。

2.2.3 パラメータ／陰関数法

一方を陰関数表現、他方をパラメータ表現式で表す。すなわち、

$$f(x,y,z)=0 \tag{2.8}$$
$$(x\ y\ z)=\mathbf{P}=\mathrm{p}(s,t) \tag{2.9}$$

ここで、s,t はパラメータである。式 2.9を式 2.8に代入すれば、次式が得られる。

$$f'(s,t)=0 \tag{2.10}$$

式 2.10を s について解き、s を t の関数として求める。すなわち、

$$s=s(t) \tag{2.11}$$

式 2.11を式 2.9に代入すれば、求める相貫曲線が次式のように t をパラメータとするパラメトリック曲線として決定される。

$$\mathbf{P}=\mathrm{p}(s(t),t)=\mathrm{p}'(t) \tag{2.12}$$

トーラスと線織面との場合にこの方法を適用する。

2.3 曲面のパラメータ表現

本節では、各曲面および直線のパラメータ表現を与える（図 2.1）。

トーラス: θ,ϕ

$$\mathbf{P}=\mathbf{P}_0+(R+r\cos\phi)(\mathrm{u}\cos\theta+\mathrm{v}\sin\theta)+\mathrm{w}r\sin\phi \tag{2.13}$$

球: θ,ϕ

$$\mathbf{P}=\mathbf{P}_0+r\sin\theta(\mathrm{i}\cos\phi+\mathrm{j}\sin\phi)+\mathrm{k}r\cos\theta \tag{2.14}$$

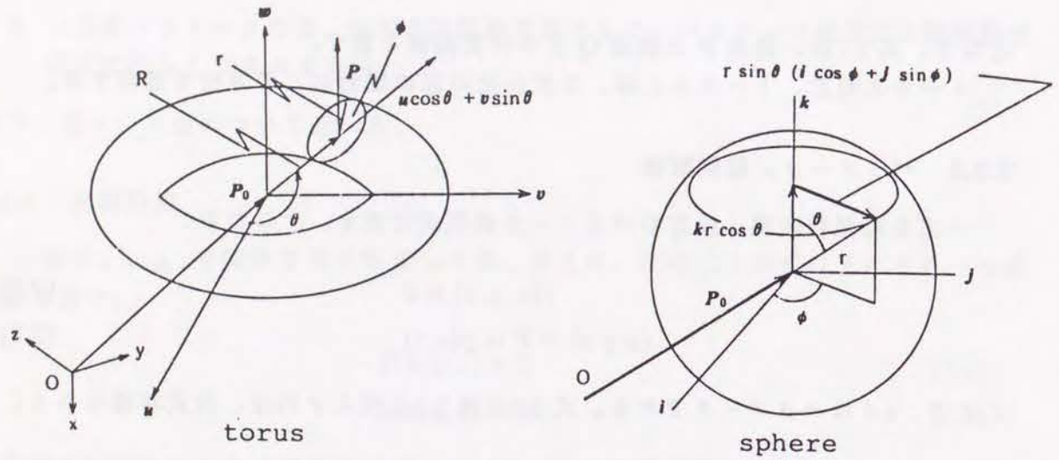


図 2.1: 曲面のパラメータ表現

円柱: θ, a

$$\mathrm{P} = \mathrm{P}_0 + r(\mathrm{u}\cos\theta + \mathrm{v}\sin\theta) + a\mathrm{w} \tag{2.15}$$

円錐: θ, a

$$\mathrm{P} = \mathrm{P}_0 + a\tan\alpha(\mathrm{u}\cos\theta + \mathrm{v}\sin\theta) + a\mathrm{w} \tag{2.16}$$

直線: k

$$\mathrm{P} = \mathrm{Q}_0 + k\mathrm{a} \tag{2.17}$$

2.4 平面と 2 次曲面との相貫曲線

$$\begin{aligned} &a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + 2a_{12}x_1x_2 + 2a_{23}x_2x_3 \\ &+2a_{31}x_3x_1 + 2a_{14}x_1 + 2a_{24}x_2 + 2a_{34}x_3 + a_{44} = 0 \end{aligned} \tag{2.18}$$

$$b_1x_1 + b_2x_2 + b_3x_3 + b_4 = 0 \tag{2.19}$$

上のように 2 次曲面と平面を表すと、その相貫曲線は次のようになる。

$$c_{jj}x_j^2 + c_{kk}x_k^2 + 2c_{jk}x_jx_k + 2c_{j4}x_j + 2c_{k4}x_k + c_{44} = 0 \tag{2.20}$$

$$b_ix_i + b_jx_j + b_kx_k + b_4 = 0 \tag{2.21}$$

$$b_i \neq 0,\ i,j,k = 1,2or3, i \neq j, j \neq k, k \neq i$$

$$\begin{aligned} c_{jj} &= a_{jj} + \frac{a_{ii}}{b_i^2}b_j^2 - 2\frac{a_{ij}}{b_i}b_j \\ c_{kk} &= a_{kk} + \frac{a_{ii}}{b_i^2}b_k^2 - 2\frac{a_{ik}}{b_i}b_k \\ c_{jk} &= a_{jk} + \frac{a_{ii}}{b_i^2}b_jb_k - \frac{a_{ij}}{b_i}b_k - \frac{a_{ik}}{b_i}b_j \\ c_{j4} &= a_{j4} + \frac{a_{ii}}{b_i^2}b_4b_j - \frac{a_{ij}}{b_i}b_4 - \frac{a_{i4}}{b_i}b_j \\ c_{k4} &= a_{k4} + \frac{a_{ii}}{b_i^2}b_4b_k - \frac{a_{ik}}{b_i}b_4 - \frac{a_{i4}}{b_i}b_k \\ c_{44} &= a_{44} + \frac{a_{ii}}{b_i^2}b_4^2 - 2\frac{a_{i4}}{b_i}b_4 \end{aligned}$$

この式は、$x_j - x_k$平面上における 2 次曲線（式 2.20）を平面（式 2.21）上に投影したものと解釈できる（図 2.2）。又、2 次曲線（式 2.20）は係数 c によって楕円、双曲線、放物線、2 直線に分類できる。

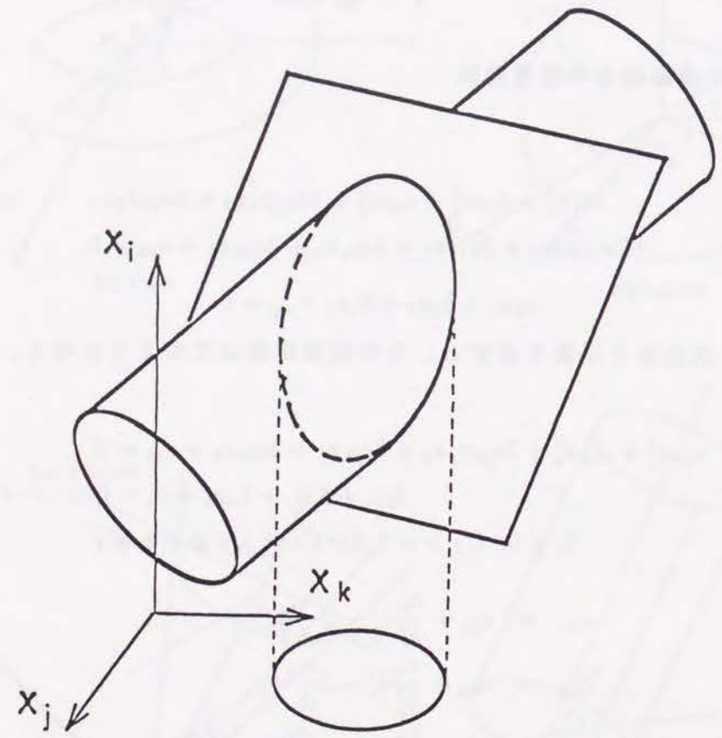


図 2.2: 平面と 2 次曲面との相貫曲線

2.5 2 次曲面同志の相貫曲線

本節では 2 次曲面同志の相貫曲線をパラメータ法によって求める手法を示す。ただし、ここで扱う 2 次曲面は、円柱、円錐、球に限定する。

いま、2 つの円柱が次のように与えられたとする。

$$\mathbf{P} = \mathbf{P}_0 + r_1(\mathbf{u}_1 \cos\theta_1 + \mathbf{v}_1 \sin\theta_1) + a_1\mathbf{w}_1 \tag{2.22}$$

$$\mathbf{Q} = \mathbf{Q}_0 + r_2(\mathbf{u}_2 \cos\theta_2 + \mathbf{v}_2 \sin\theta_2) + a_2\mathbf{w}_2 \tag{2.23}$$

マトリクス $E = [\mathbf{u}_2\ \mathbf{v}_2\ \mathbf{w}_2]^T$ を各ベクトルの左側から作用させて式 2.22,2.23のベクトルを回転変換すると次の 2 式が得られる。

$$\mathbf{P}' = \mathbf{P}'_0 + r_1(\mathbf{u} \cos\theta_1 + \mathbf{v} \sin\theta_1) + a_1\mathbf{w} \tag{2.24}$$

$$\mathbf{Q}' = \mathbf{Q}'_0 + r_2(\mathbf{i} \cos\theta_2 + \mathbf{j} \sin\theta_2) + a_2\mathbf{k} \tag{2.25}$$

ただし、$\mathbf{i},\mathbf{j},\mathbf{k}$ はそれぞれ x_1, x_2, x_3 方向の単位ベクトル、$\mathbf{P}'_0 = E\mathbf{P}_0$、$\mathbf{Q}'_0 = E\mathbf{Q}_0$、$\mathbf{u} = E\mathbf{u}_1$、$\mathbf{v} = E\mathbf{v}_1$、$\mathbf{w} = E\mathbf{w}_1$ である。

$\mathbf{P}' = \mathbf{Q}'$ より x_1, x_2, x_3 成分の 3 個の式が得られるが、その 3 個の式より 2 番目の円柱に関するパラメータ θ_2, a_2 を消去すると、次のようにパラメータ a_1 に関する 2 次方程式が得られる。

$$Aa_1^2 + 2Ba_1 + C = 0 \tag{2.26}$$

$$\begin{aligned} A &= w_1^2 + w_2^2 \\ B &= w_1 f_1 + w_2 f_2 \\ C &= f_1^2 + f_2^2 - r_2^2 \\ \mathbf{f} &= r_1(\mathbf{u} \cos\theta_1 + \mathbf{v} \sin\theta_1) + \mathbf{T} \\ \mathbf{T} &= \mathbf{P}'_0 - \mathbf{Q}'_0 \end{aligned}$$

従って、係数 A, B, C は θ_1 の関数又は定数となり、式 2.26を解くことにより a_1 は θ_1 の関数として求まる。その a_1 を式 2.22に代入すれば、$\mathbf{P}$ はパラメータ θ_1 のみの関数となり、これが求める相貫曲線の式となる。

又、式 2.26の判別式 D は以下のように θ_1 の関数となる。

$$\begin{aligned} D &= B^2 - AC \\ &= -(w_1 f_2 - w_2 f_1)^2 + Ar_2^2 \\ &= -\{r_1(-u_3 \sin\theta_1 + v_3 \cos\theta_1) + w_1 t_2 - w_2 t_1\}^2 + Ar_2^2 \\ &= -\{B_1 \sin(\theta_1 + \phi) - B_2\}^2 + B_3 \end{aligned} \tag{2.27}$$

ただし、

$$\begin{aligned} B_1 &= r_1\sqrt{u_3^2+v_3^2} \neq 0 \\ B_2 &= w_2t_1 - w_1t_2 \\ B_3 &= Ar_2^2 \\ \phi &= \tan^{-1}(-v_3/u_3) \end{aligned}$$

式2.26の解a_1が存在するのは判別式が正又は零の場合であるから、$D \geq 0$をθ_1について解くことにより、解の存在するθ_1の区間、すなわち、相貫曲線存在区間を求めることができる。式2.27より円柱と円柱との相貫曲線存在区間は、$X = (B_2 - \sqrt{B_3})/B_1, Y = (B_2 + \sqrt{B_3})/B_1$とすれば以下のようになる。

1. $Y < -1$ または $1 < X$のとき、解なし（相貫曲線なし）。

2. $X \leq -1$ かつ $1 \leq Y$のとき、全区間。

3. $-1 < X \leq 1$ かつ $1 \leq Y$のとき、

$$\sin^{-1} X - \phi \leq \theta_1 \leq \pi - \sin^{-1} X - \phi.$$

4. $X \leq -1$ かつ$-1 \leq Y < 1$のとき、

$$-\pi - \sin^{-1} Y - \phi \leq \theta_1 \leq \sin^{-1} Y - \phi.$$

5. $-1 < X \leq 1$ かつ$-1 \leq Y < 1$のとき、

$$\begin{aligned} \sin^{-1} X - \phi \leq\ & \theta_1 \ \leq \sin^{-1} Y - \phi, \\ \pi - \sin^{-1} Y - \phi \leq\ & \theta_1 \ \leq \pi - \sin^{-1} X - \phi. \end{aligned}$$

他の組合せについても同様に2次方程式の形式で求まる。各係数A, B, Cを図2.3に示す。又、相貫曲線の出力例を図2.4に示す。なお、球と球の場合はパラメータ表現を用いなくても容易に求まるので省略する。

2.6 トーラスと2次曲面との相貫曲線

本節では、トーラスと球、トーラスと円柱、トーラスと円錐の相貫曲線解析解を示す。トーラスと球の場合はパラメータ法を利用する。トーラスと円柱・円錐の場合は、円柱・円錐を直線の集合と見なして、直線をパラメータ表現、トーラスを陰関数表現とするパラメータ／陰関数法を利用する。

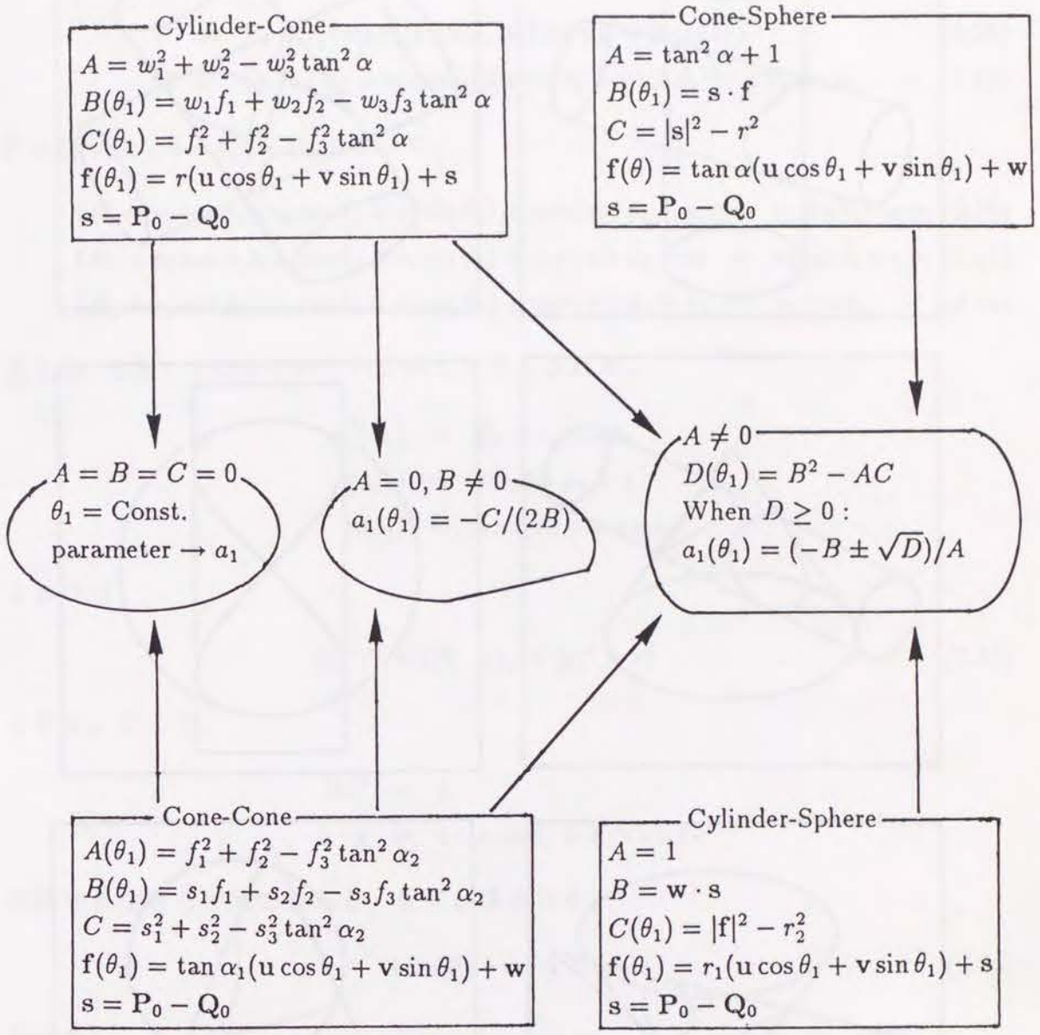


図 2.3: 2次曲面相貫曲線式の係数

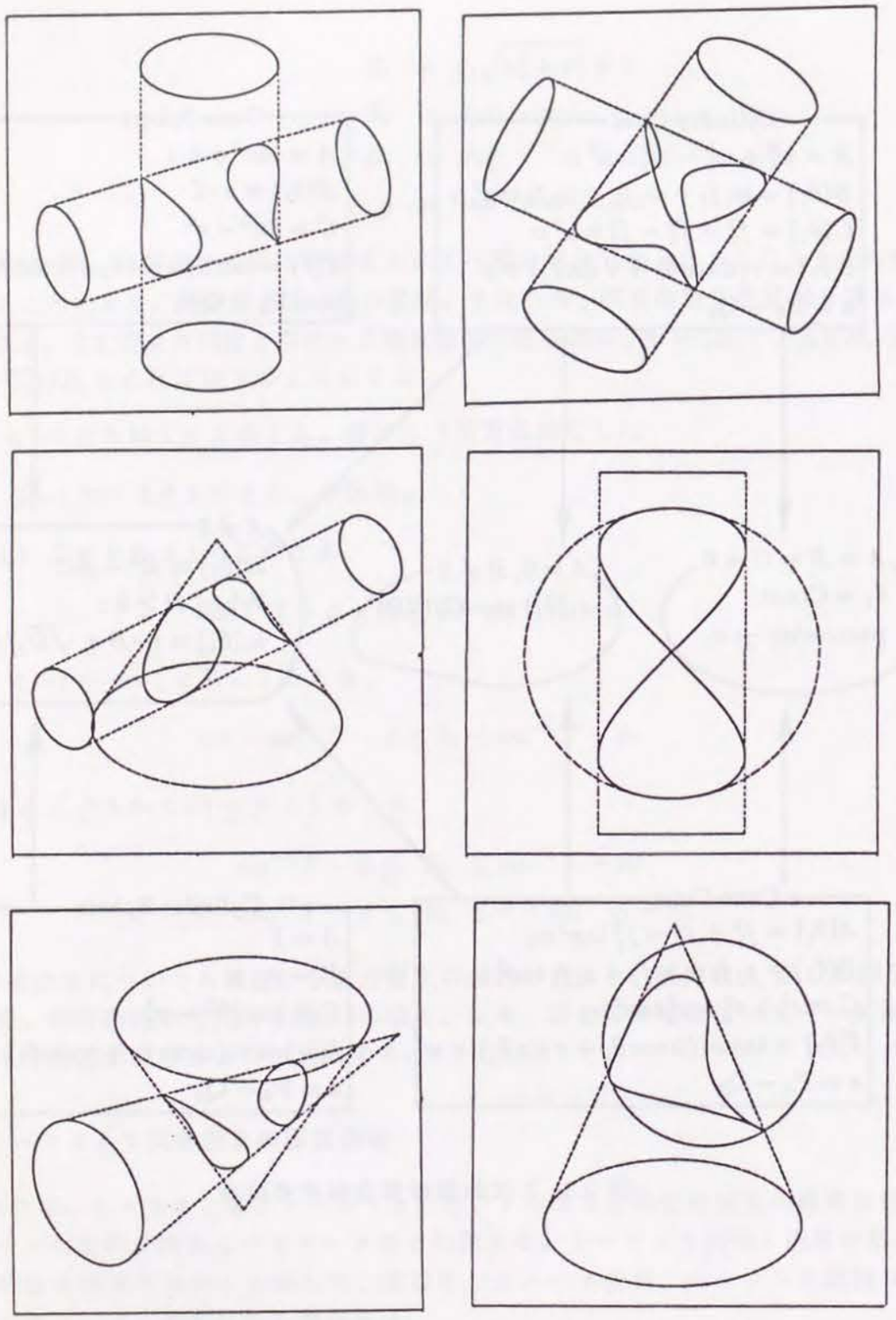


図 2.4: 2 次曲面相貫曲線の出力例

2.6.1 トーラスと球との相貫曲線

球とトーラスを次のように表す。すなわち、

$$\mathbf{P} = \mathbf{P}_0 + r_1\{\sin\theta_1(\mathbf{i}\cos\phi_1 + \mathbf{j}\sin\phi_1) + \mathbf{k}\cos\theta_1\} \quad (2.28)$$

$$\mathbf{Q} = \mathbf{Q}_0 + (R_2 + r_2\cos\phi_2)(\mathbf{u}\cos\theta_2 + \mathbf{v}\sin\theta_2) + \mathbf{w}r_2\sin\phi_2 \quad (2.29)$$

$\mathbf{P} = \mathbf{Q}$ より、$\mathbf{s} = \mathbf{Q}_0 - \mathbf{P}_0$ とおいて、

$$(R_2 + r_2\cos\phi_2)(u_1\cos\theta_2 + v_1\sin\theta_2) + w_1r_2\sin\phi_2 + s_1 = r_1\sin\theta_1\cos\phi_1 \quad (2.30)$$

$$(R_2 + r_2\cos\phi_2)(u_2\cos\theta_2 + v_2\sin\theta_2) + w_2r_2\sin\phi_2 + s_2 = r_1\sin\theta_1\sin\phi_1 \quad (2.31)$$

$$(R_2 + r_2\cos\phi_2)(u_3\cos\theta_2 + v_3\sin\theta_2) + w_3r_2\sin\phi_2 + s_3 = r_1\cos\theta_1 \quad (2.32)$$

式 2.30、2.31、2.32をそれぞれ 2 乗して足し合わせ、

$$\begin{aligned} f(\phi_2) &= R_2 + r_2\cos\phi_2 \\ \mathrm{g}(\phi_2) &= \mathbf{w}r_2\sin\phi_2 + \mathbf{s} \\ \mathrm{h}(\theta_2) &= \mathbf{u}\cos\theta_2 + \mathbf{v}\sin\theta_2 \end{aligned}$$

とおけば、

$$|\mathrm{h}|^2 f^2 + 2(\mathrm{h}\cdot\mathrm{g})f + |\mathrm{g}|^2 = r_1^2 \quad (2.33)$$

となる。ここで、

$$\begin{aligned} |\mathrm{h}|^2 &= 1 \\ \mathrm{h}\cdot\mathrm{g} &= \mathbf{s}\cdot\mathbf{u}\cos\theta_2 + \mathbf{s}\cdot\mathbf{v}\sin\theta_2 \end{aligned}$$

の関係に注意してまとめると、次式が得られる。

$$A\sin(\theta_2 + \alpha) = B(\phi_2) \quad (2.34)$$

ただし、

$$\begin{aligned} A &= \sqrt{(\mathbf{s}\cdot\mathbf{u})^2 + (\mathbf{s}\cdot\mathbf{v})^2} \\ \alpha &= \tan^{-1}\left(\frac{\mathbf{s}\cdot\mathbf{u}}{\mathbf{s}\cdot\mathbf{v}}\right) \\ B(\phi_2) &= \frac{r_1^2 - f^2 - |\mathrm{g}|^2}{2f} \end{aligned}$$

1. $A \neq 0$ の場合

式 2.34より、θ_2はϕ_2の関数として求まる。すなわち、$|B/A| \leq 1$ の範囲で、

$$\theta_2 = \sin^{-1}(\frac{B}{A}) - \alpha \tag{2.35}$$

式 2.35を式 2.29に代入すれば、$\mathbf{Q}$ はϕ_2のみの関数となり、これが求める相貫曲線の式となる。またこの場合、$|B/A| \leq 1$ となるϕ_2の区間、すなわち、相貫曲線存在区間は以下のように求まる。

(a) $B_2/A_2 > 1$ または $B_3/A_3 < -1$ または $B_2/A_2 > B_3/A_3$のとき、
相貫曲線なし。

(b) $B_2/A_2 \leq -1$ かつ $B_3/A_3 \geq 1$ のとき、
$0 \leq \phi_2 \leq 2\pi$（全区間）

(c) $B_2/A_2 \leq -1$ かつ $-1 \leq B_3/A_3 < 1$ のとき、
$\pi - \sin^{-1}(B_3/A_3) - \alpha_3 \leq \phi_2 \leq 2\pi + \sin^{-1}(B_3/A_3) - \alpha_3$

(d) $B_3/A_3 \geq 1$ かつ $-1 < B_2/A_2$ のとき、
$\sin^{-1}(B_2/A_2) - \alpha_2 \leq \phi_2 \leq \pi - \sin^{-1}(B_2/A_2) - \alpha_2$

(e) $B_2/A_2 > -1$ かつ $B_3/A_3 < 1$ のとき、
$\sin^{-1}(B_2/A_2) - \alpha_2 \leq \phi_2 \leq \pi - \sin^{-1}(B_2/A_2) - \alpha_2$
かつ、
$\pi - \sin^{-1}(B_3/A_3) - \alpha_3 \leq \phi_2 \leq 2\pi + \sin^{-1}(B_3/A_3) - \alpha_3$

ただし、

$$\begin{aligned}
A_2 &= 2r_2\sqrt{(A+R_2)^2 + (\mathbf{w}\cdot\mathbf{s})^2} \\
\alpha_2 &= \tan^{-1}(\frac{A+R_2}{\mathbf{w}\cdot\mathbf{s}}) \\
B_2 &= r_1^2 - (R_2^2 + r_2^2 + |\mathbf{s}|^2 + 2AR_2) \\
A_3 &= 2r_2\sqrt{(R_2-A)^2 + (\mathbf{w}\cdot\mathbf{s})^2} \\
\alpha_3 &= \tan^{-1}(\frac{R_2-A}{\mathbf{w}\cdot\mathbf{s}}) \\
B_3 &= r_1^2 - (R_2^2 + r_2^2 + |\mathbf{s}|^2 - 2AR_2)
\end{aligned}$$

2. $A = 0$ の場合、

この場合は、θ_2がϕ_2の関数としては求まらないが、$B = 0$ という条件より$\phi_2 =$ *const.*の形で求まる。すなわち、

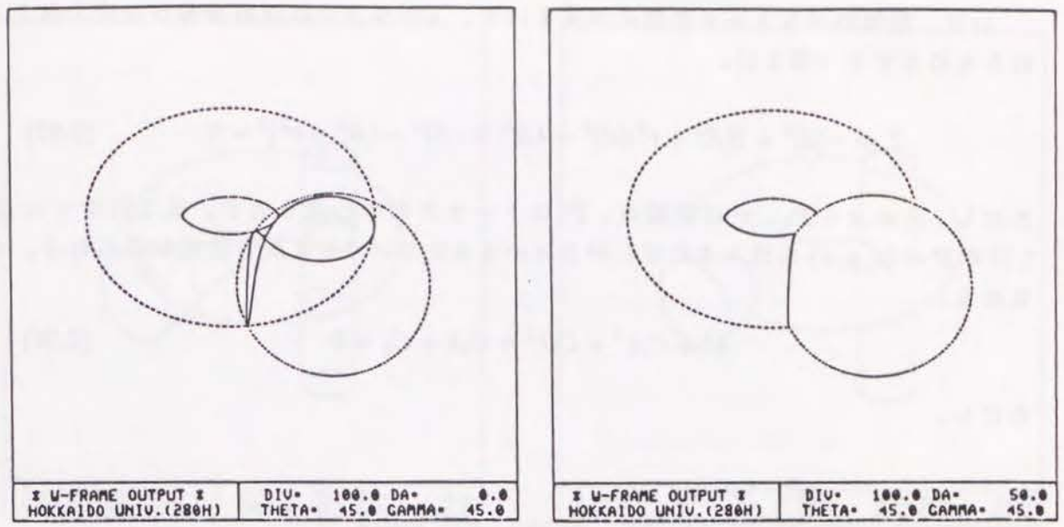


図 2.5: トーラスと球との相貫曲線

(a) $|B_1/A_1| \leq 1$ のとき、$0 \leq \theta_2 \leq 2\pi$の区間で、

$$\phi_2 = \sin^{-1}(B_1/A_1) - \alpha_1 \tag{2.36}$$

(b) $|B_1/A_1| > 1$ のとき、相貫曲線なし。

ただし、

$$\begin{aligned}
A_1 &= 2r_2\sqrt{R_2^2 + (\mathbf{w}\cdot\mathbf{s})^2} \\
\alpha_1 &= \tan^{-1}(R_2/\mathbf{w}\cdot\mathbf{s}) \\
B_1 &= r_1^2 - (R_2^2 + r_2^2 + |\mathbf{s}|^2)
\end{aligned}$$

式 2.36を式 2.29に代入すれば $\mathbf{Q}$ はθ_2のみの関数となり、これが相貫曲線の式となる。

出力結果を図 2.5に示す。

2.6.2 トーラスと線織面との相貫曲線

線織面は、直線の集合と考えることができる。従って、直線とトーラスとの相貫点を求める式が計算できれば、トーラスと線織面との相貫曲線を与えることができる。

いま、直線のパラメータ表現式が式 2.17で、トーラスの陰関数表現が次式で与えられるものとする（図 2.1）。

$$T = -|\mathrm{d}|^4 + 2(R^2+r^2)|\mathrm{d}|^2 - 4R^2(\mathrm{w}\cdot\mathrm{d})^2 - (R^2-r^2)^2 = 0 \qquad (2.37)$$

ただし、$\mathrm{d} = \mathrm{x} - \mathrm{P}_0$、x は空間点、$\mathrm{P}_0$はトーラスの中心点を表す。式 2.37の x に式 2.17の $\mathrm{P} = (x, y, z)$ を代入すれば、パラメータ kについての 4 次方程式が得られる。すなわち、

$$k^4 + C_1k^3 + C_2k^2 + C_3k + C_4 = 0 \qquad (2.38)$$

ただし、

$$\begin{aligned} C_1 &= 2b_1 \\ C_2 &= b_1^2 + 2b_2 - 2(R^2+r^2) + 4R^2b_3^2 \\ C_3 &= 2b_1b_2 - 2b_1(R^2+r^2) + 8R^2b_3b_4 \\ C_4 &= b_2^2 - 2(R^2+r^2)b_2 + 4R^2b_4^2 + (R^2-r^2)^2 \\ b_1 &= 2\mathrm{a}\cdot(\mathrm{Q}_0 - \mathrm{P}_0) \\ b_2 &= |\mathrm{Q}_0 - \mathrm{P}_0|^2 \\ b_3 &= \mathrm{w}\cdot\mathrm{a} \\ b_4 &= \mathrm{w}\cdot(\mathrm{Q}_0 - \mathrm{P}_0) \end{aligned}$$

式 2.38を公式を使って解き、得られた kを式 2.17に代入すれば、求める相貫点 $\mathrm{P} = (x, y, z)$ が得られる。

以上で、トーラスと直線との相貫点が得られたから、直線から作られる曲面、すなわち、線織面とトーラスとの相貫曲線は、各母線の始点 Q_0と方向余弦 a をパラメータ表現できれば求まることになる。ここでは、円柱と円錐の Q_0、a を示す（図 2.1）。

$$\begin{aligned} \text{円柱:}\ \mathrm{Q}_0 &= \mathrm{P}_0 + r(\mathrm{u}\cos\theta + \mathrm{v}\sin\theta) && (2.39) \\ \mathrm{a} &= \mathrm{w} && (2.40) \\ \text{円錐:}\ \mathrm{Q}_0 &= \mathrm{P}_0 && (2.41) \\ \mathrm{a} &= \tan\alpha(\mathrm{u}\cos\theta + \mathrm{v}\sin\theta) + \mathrm{w} && (2.42) \end{aligned}$$

パラメータはともにθである。出力結果を図 2.6に示す。

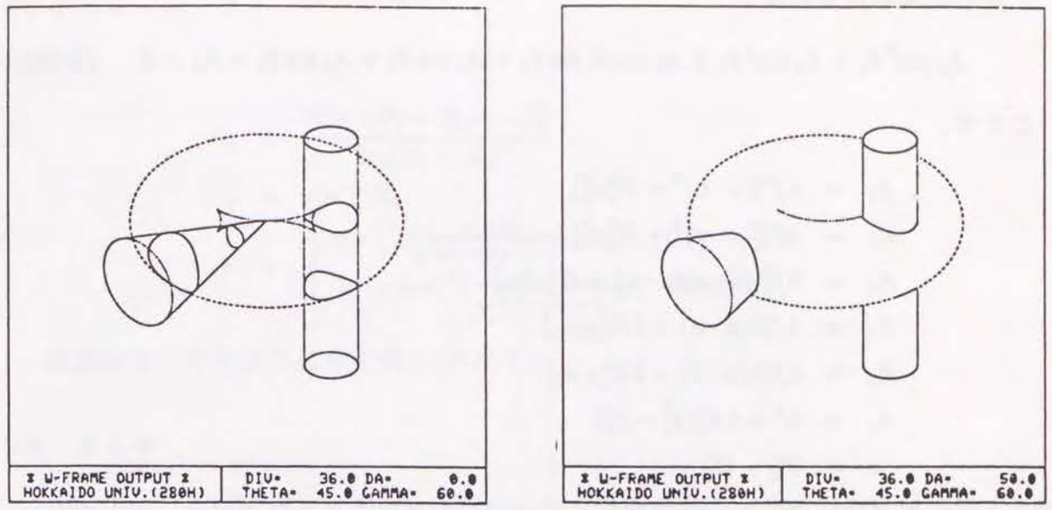


図 2.6: トーラスと線織面との相貫曲線

2.7 トーラス同志の相貫曲線

本節では、4 次曲面であるトーラス同志の相貫曲線解析解をパラメータ法を使って求める。二つのトーラスが次のように与えられたとする。

$$\begin{aligned} \mathrm{P} &= \mathrm{P}_0 + (R_1 + r_1\cos\phi_1)(\mathrm{u}_1\cos\theta_1 + \mathrm{v}_1\sin\theta_1) + \mathrm{w}_1r_1\sin\phi_1 && (2.43) \\ \mathrm{Q} &= \mathrm{Q}_0 + (R_2 + r_2\cos\phi_2)(\mathrm{u}_2\cos\theta_2 + \mathrm{v}_2\sin\theta_2) + \mathrm{w}_2r_2\sin\phi_2 && (2.44) \end{aligned}$$

まず始めにマトリックス Eを次のように作る。

$$E = \begin{bmatrix} \mathrm{u}_2 \\ \mathrm{v}_2 \\ \mathrm{w}_2 \end{bmatrix} \qquad (2.45)$$

このマトリックス Eを使って、式 2.43、2.44の各ベクトルを回転変換すると、次式を得る。

$$\begin{aligned} \mathrm{P}' &= \mathrm{P}'_0 + (R_1 + r_1\cos\phi_1)(\mathrm{u}\cos\theta_1 + \mathrm{v}\sin\theta_1) + \mathrm{w}r_1\sin\phi_1 && (2.46) \\ \mathrm{Q}' &= \mathrm{Q}'_0 + (R_2 + r_2\cos\phi_2)(\mathrm{i}\cos\theta_2 + \mathrm{j}\sin\theta_2) + \mathrm{k}r_2\sin\phi_2 && (2.47) \end{aligned}$$

ここで、i,j,k はそれぞれ x, y, z軸方向の単位ベクトル、$\mathrm{P}'_0 = E\mathrm{P}_0$、$\mathrm{Q}'_0 = E\mathrm{Q}_0$、$\mathrm{u} = E\mathrm{u}_1$、$\mathrm{v} = E\mathrm{v}_1$、$\mathrm{w} = E\mathrm{w}_1$ である。式 2.46,2.47を連立させてパラメータϕ_2, θ_2を消去

すると次式が得られる。

$$A_1\cos^2\theta_1 + A_2\sin^2\theta_1 + A_3\cos\theta_1\sin\theta_1 + A_4\cos\theta_1 + A_5\sin\theta_1 + A_6 = 0 \qquad (2.48)$$

ここで、

$$\begin{aligned}
A_1 &= 4f^2[(\mathbf{s}\cdot\mathbf{u})^2 + R_2^2u_3^2] \\
A_2 &= 4f^2[(\mathbf{s}\cdot\mathbf{v})^2 + R_2^2v_3^2] \\
A_3 &= 8f^2[(\mathbf{s}\cdot\mathbf{u})(\mathbf{s}\cdot\mathbf{v}) + R_2^2u_3v_3] \\
A_4 &= 4f[h(\mathbf{s}\cdot\mathbf{u}) + 2R_2^2g_3u_3] \\
A_5 &= 4f[h(\mathbf{s}\cdot\mathbf{v}) + 2R_2^2g_3v_3] \\
A_6 &= h^2 + 4R_2^2(g_3^2 - r_2^2) \\
\mathbf{s} &= \mathbf{P}_0' - \mathbf{Q}_0' \\
\mathbf{f} &= R_1 + r_1\cos\phi_1 \\
\mathbf{g} &= \mathbf{w}r_1\sin\phi_1 + \mathbf{s} \\
h &= 2r_1(\mathbf{w}\cdot\mathbf{s}\sin\phi_1 + R_1\cos\phi_1) + R_1^2 + r_1^2 - R_2^2 - r_2^2 + |\mathbf{s}|^2
\end{aligned}$$

である。従って、$A_i(i = 1, 2, \cdots, 6)$ はϕ_1の関数である。さらに、$cos\theta_1 = \pm\sqrt{1-\sin^2\theta_1}$ の関係を式 2.48に代入すれば、次式のような$\sin\theta_1$に関する 4 次方程式が得られる。

$$B_0\sin^4\theta_1 + B_1\sin^3\theta_1 + B_2\sin^2\theta_1 + B_3\sin\theta_1 + B_4 = 0 \qquad (2.49)$$

ただし、

$$\begin{aligned}
B_0 &= (A_2 - A_1)^2 + A_3^2 \\
B_1 &= 2[(A_2 - A_1)A_5 + A_3A_4] \\
B_2 &= A_5^2 + 2(A_2 - A_1)(A_1 + A_6) - A_3^2 + A_4^2 \\
B_3 &= 2[A_5(A_1 + A_6) - A_3A_4] \\
B_4 &= (A_1 + A_6)^2 - A_4^2
\end{aligned}$$

従って、式 2.49を公式を使って解けば、θ_1がϕ_1の関数として求まるので、それを式 2.43に代入したものが相貫曲線の式となる。ただし、$u_3 = v_3 = \mathbf{s}\cdot\mathbf{u} = \mathbf{s}\cdot\mathbf{v} = 0$ の場合、すなわち、二つのトーラスの軸 $\mathbf{w}_1, \mathbf{w}_2$が一致している場合は、$\theta_1$を$\phi_1$の関数として求めることができず、$\phi_1 = const.$の形で解が求まる。すなわち、

(i) $|R_3| \le 1$ のとき、 $\phi_1 = \beta + \gamma$

(ii) $|R_3| > 1$ のとき、 相貫曲線なし

ただし、

$$\begin{aligned}
R_3 &= \frac{r_1^2 + (R_1 - R_2)^2 - r_2^2}{2r_1|R_1 - R_2|} \\
\gamma &= \cos^{-1} R_3 \\
\beta &= \begin{cases} \cos^{-1}\left(\frac{R_2 - R_1}{\sqrt{|\mathbf{s}|^2 + (R_2 - R_1)^2}}\right) & \text{if } \mathbf{s}\cdot\mathbf{w} \le 0 \\ 2\pi - \cos^{-1}\left(\frac{R_2 - R_1}{\sqrt{|\mathbf{s}|^2 + (R_2 - R_1)^2}}\right) & \text{otherwise} \end{cases}
\end{aligned}$$

相貫曲線の図形出力結果を図 2.7に示す。

2.8 まとめ

本章では、形状モデリングにおいて中心的な問題であるところの曲面間の相貫曲線を解析的に求める手法について論じた。すなわち、

1. 曲面間の相貫曲線解析解の一般的な求め方として、陰関数法、パラメータ法およびパラメータ／陰関数法を論じた。
2. 平面と 2 次曲面との相貫曲線解析解を陰関数法で求めた。
3. 2 次曲面同志の相貫曲線解析解をパラメータ法で求めた。
4. トーラスと球との相貫曲線解析解をパラメータ法で求めた。
5. トーラスと線織面との相貫曲線解析解をパラメータ／陰関数法で求めた。
6. トーラス同志の相貫曲線解析解をパラメータ法で求めた。

なお、パラメータ法、パラメータ／陰関数法においては、相貫曲線が存在するパラメータの区間、すなわち、相貫曲線存在区間が得られることが重要な場合がある。これについては、上記のすべての組合せについて解析的には求めることができず、トーラスと線織面、トーラス同志の場合について解決されていない。

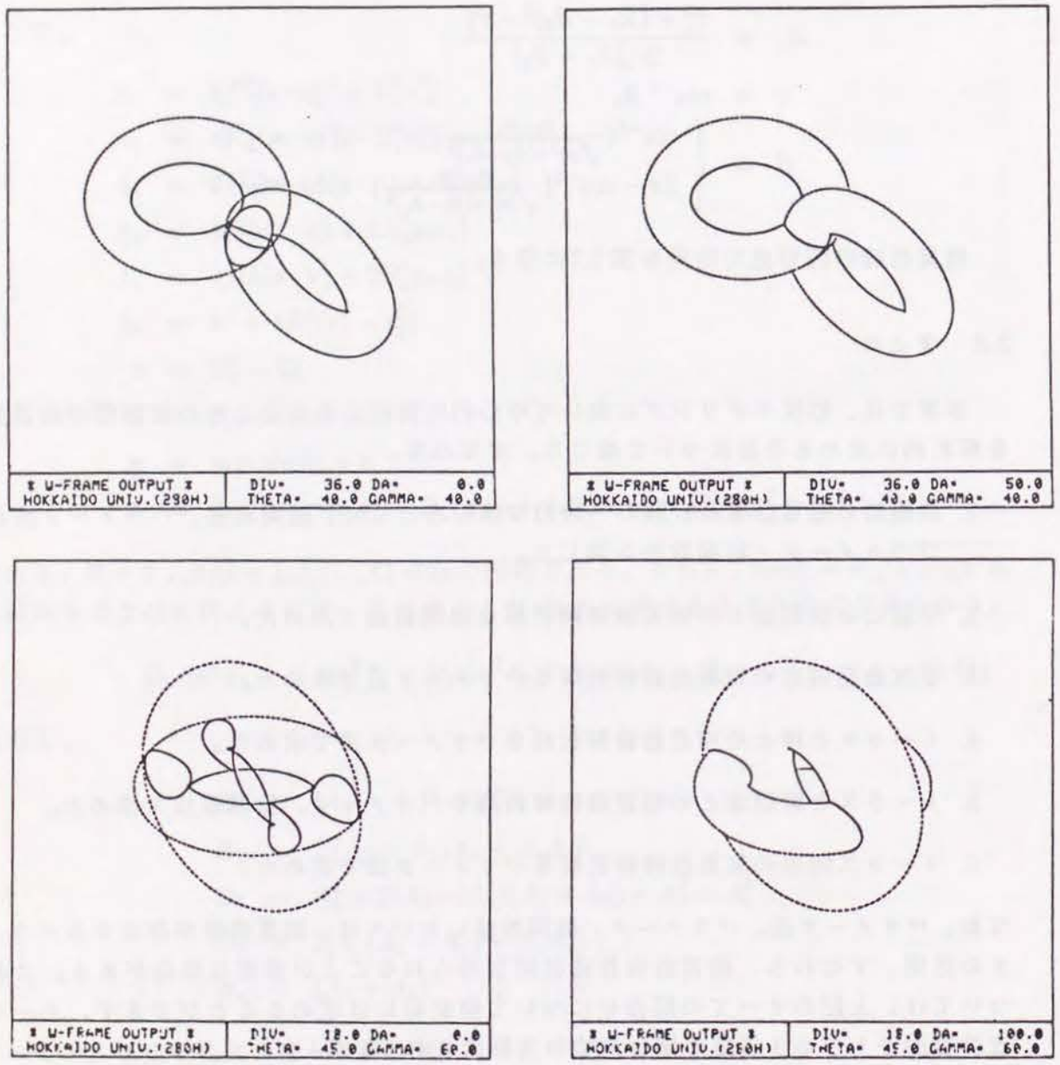


図 2.7: トーラス同志の相貫曲線

参考文献

[1] Weiss, Ruth A. : "BE VISION, A Package of IBM 7090 Fortran Program to Draw Orthographic Views of Plane and Quadric Surfaces", J. ACM, Vol. 13, No. 2, Apr. (1966)194.

[2] Joshua Levin : "A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces", Com. ACM, Vol.19, No.10, Oct. (1976)555.

[3] 沖野教郎、嘉数侑昇、久保　洋：自動設計プロセサ TIPS-1 の開発、精密機械、44, 3(1978) 371.

[4] 嘉数侑昇、沖野教郎、渡部広一：２次曲面及び４次曲面（トーラス）の相貫曲線解析解、北海道大学工学部研究報告、第 120 号（ 1984 ）65.

[5] 渡部広一、嘉数侑昇、沖野教郎: ソリッド形状モデリングにおける CSG から B-Reps への解析的変換の研究、精密工学会誌、**53** 、2(1987)315.

第 3 章

フィレットの CSG 表現法

3.1 はじめに

序論で述べたように、CSG 表現 [1,2] では、プリミティブと呼ばれる比較的単純な立体を組み合わせて形状を定義する。しかし、そのような単純な立体では定義しきれない場合がある。多くの工業製品を見ると大まかにはプリミティブの組合せで表現できるが、細かくみると、いたる所に丸みが付いており、この丸み付けの表現が問題となる。この丸み部分のことをフィレットと呼んでおり、本章では、CSG でいかにフィレットを表現するかという問題を扱い、CSG の高精度化を図る。

直線や円弧に沿う部分のフィレットは、円柱やトーラスといったプリミティブで表現可能であるので、ここではそれ以外の曲線に沿うフィレットについて考え、特に 2 次曲面相貫曲線に沿うフィレットを扱う。2 次曲面相貫曲線は、第 2 章で述べたように、パラメトリック曲線となるので、本章で述べる手法は、他のパラメトリック曲線の場合にも適用可能であると思われる。

フィレットを CSG で表現するためには、フィレットプリミティブを準備しなくてはならない。本研究では CSG モデラとして TIPS-1[7] を利用し、その CONTOR と呼ばれる任意形状プリミティブの定義機能を使ってフィレットプリミティブを作成する。CONTOR は目的の形状に対するペナルティ関数 [6] を計算するサブルーチンを定義することにより利用できる。ペナルティ関数は、空間中の任意の 1 点が与えられたとき、その点が形状表面にあればゼロ、内部にあれば正、外部にあれば負の値を返す関数である。このとき、形状表面から離れるほどその値の絶対値が大きくなるようにしておかなくてはならない。したがって、CONTOR を使ってフィレットを定義するには、目的のフィレット形状に対するペナルティ関数を準備すれば良いことになる。

ここでは、フィレットに対してペナルティ値を計算する手続きとして、スウィープ

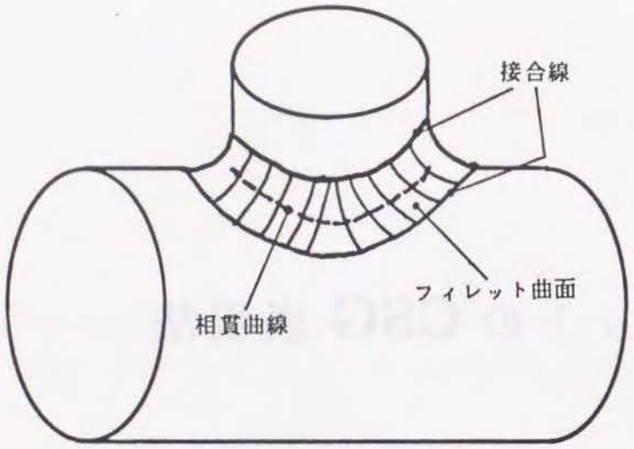


図 3.1: フィレットの構成

オペレーション[5]の適用を図った。すなわち、ある2次元断面形状を2次曲面相貫曲線に沿ってスウィープすることによりフィレットの生成を行った。ただし、本章で扱う2次曲面は円柱、円錐と球の3種類である。

3.2 フィレットの構成

図3.1に示すように、フィレットはフィレット曲面およびそれを仕切る接合線により構成される。フィレット曲面は、凸形、平形、凹形があり、その曲面形状は断面でみるとそれぞれ凸円弧、直線、凹円弧であるが、全体的には相貫曲線に沿って複雑な曲面となる。

3.3 フィレット創成の手順

今、図3.2に示すような結合する二つの2次曲面C_1、C_2を想定する。3次元直交座標系で表される空間（Xと記す）においてC_1、C_2が式3.1の関数で記述されるとする。

$$\left.\begin{array}{ll} C_1: & g(\mathbf{X})=0 \\ C_2: & h(\mathbf{X})=0 \end{array}\right\} \tag{3.1}$$

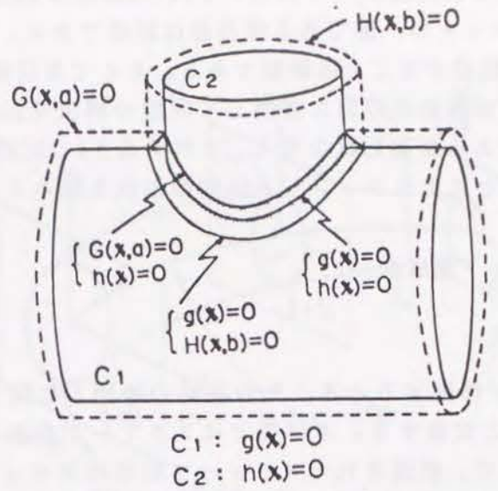


図 3.2: フィレット曲面と2次曲面との接合条件

また、2次曲面C_1、C_2をその曲面の法線方向へオフセットした、オフセット2次曲面C_{1off}、C_{2off}は式3.2のように表されるものとする。

$$\left.\begin{array}{ll} C_{1off}: & G(\mathbf{X},a)=0 \\ C_{2off}: & H(\mathbf{X},b)=0 \end{array}\right\} \tag{3.2}$$

ここで、変数a,bはオフセット値とする。オフセット値がゼロの場合、オフセット2次曲面は当然、元の2次曲面に一致するので次式が成立する。

$$\left.\begin{array}{l} G(\mathbf{X},0)=g(\mathbf{X}) \\ H(\mathbf{X},0)=h(\mathbf{X}) \end{array}\right\} \tag{3.3}$$

2次曲面C_1、C_2間の相貫曲線は次の連立方程式を解いて得られる。

$$\left.\begin{array}{l} g(\mathbf{X})=0 \\ h(\mathbf{X})=0 \end{array}\right\} \tag{3.4}$$

オフセット2次曲面C_{1off}（またはC_{2off}）を用いてフィレット曲面と2次曲面C_2（またはC_1）との接合線を指定するものとすれば、接合線は次式を解いて得られる。

$$\left.\begin{array}{l} G(\mathbf{X},a)=0 \\ h(\mathbf{X})=0 \end{array}\right\} \tag{3.5}$$

$$\left.\begin{array}{l} g(\mathbf{X})=0 \\ H(\mathbf{X},b)=0 \end{array}\right\} \tag{3.6}$$

すなわち、式3.5の解は2次曲面C_2上、同様に、式3.6の解はC_1上の接合線を意味する。

以上のようにフィレットの一部である接合線は記述できた。しかし、フィレット曲面そのものを関係式で記述することは困難である。そこで本研究では、フィレット曲面創成問題をスウィープ曲面創成問題に置換して問題の解決を図る。すなわち、式3.5、3.6の接合条件を満たすある断面形状を考え、これを式3.4で記述される相貫曲線に沿ってスウィープさせることによりフィレット曲面の創成を試みる。

3.4 スウィープソリッド創成法

3.4.1 スウィープ方法

ある2次元閉曲線が移動するとき、その曲線の時間tに関する軌跡が作る曲面をスウィープ曲面[2],[5]と定義する。本研究ではスウィープ曲面の表面のみならず内部をも処理対象とするので、創成されるスウィープ形状はスウィープソリッド（Swept Volume）と呼ぶことにする。本研究でのスウィープ方法は次のように記述できる。

1. 被スウィープ形状は3次元直交座標系で表される空間に設定された適当な平面X_S-Y_S(ただし、X_S、Y_Sはその平面の座標軸）に定義されるとする。
2. 被スウィープ形状はいくつかの曲線分によって構成される閉曲線、または、単一の閉曲線で表現されるものとし、移動の際、形状の大きさや形そのものが時間tとともに変化することを許す。
3. 被スウィープ形状上の適当な1点C_Pの移動軌跡を移動軸（以下、スウィープ軸と呼ぶ）として与える。本研究の場合、2次曲面間の相貫曲線をスウィープ軸として用いることができる。

以上述べたスウィープ方法は図3.3のように示される。

3.4.2 被スウィープ形状定義平面の設定

平面X_S-Y_Sを設定するための入力情報は任意の空間点S_Pの座標値のみである（ペナルティ関数の設定のため)。平面X_S-Y_Sは次のように設定される（図3.4)。

1. 空間点S_Pから一方の2次曲面へ垂線を下ろし中心軸Wとの交点A_Pを求める。ただし、空間点S_Pは垂線を下ろす対象2次曲面の中心軸W上には存在しないものとする。(球の場合はA_Pは球の中心点となる。)
2. 点A_Pを含み中心軸Wに垂直な平面$U-V$を設定する。

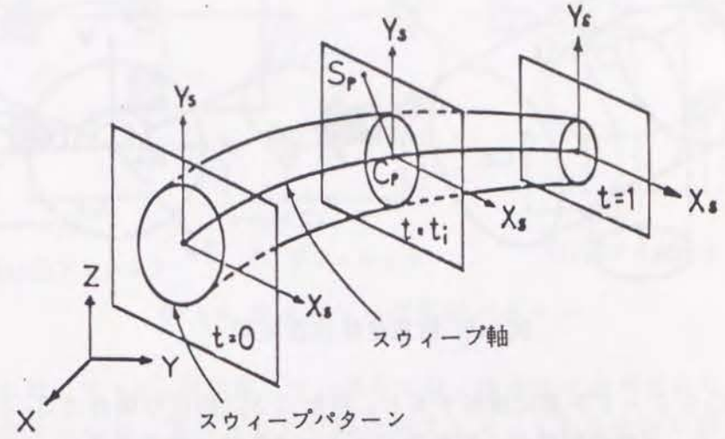


図 3.3: 被スウィープ形状とスウィープ軸の定義

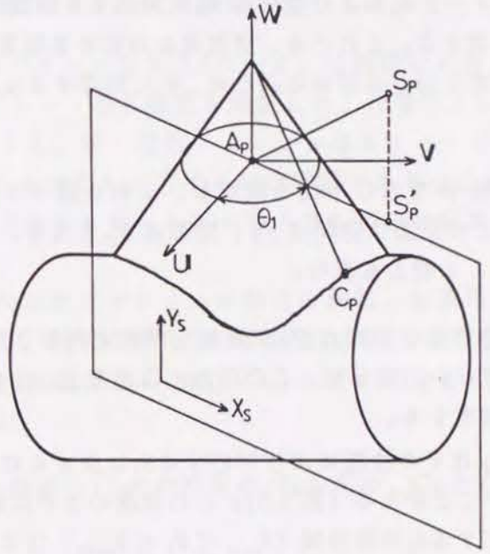


図 3.4: スウィープ形状定義平面

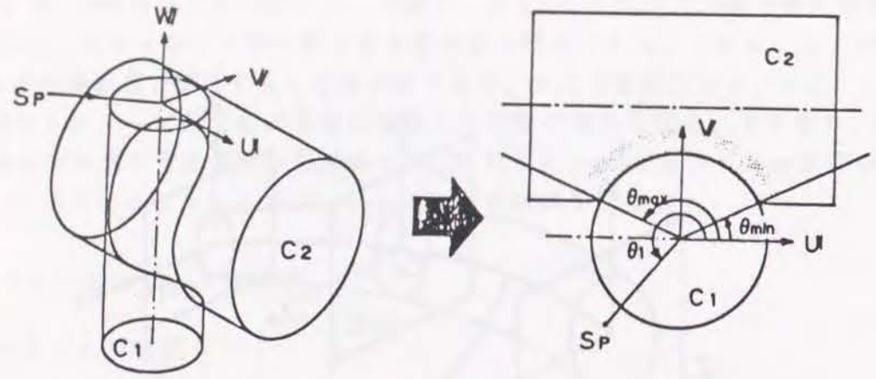


図 3.5: 相貫曲線存在区間

3. 空間点S_Pを$U-V$平面に投影すると、投影点S'_P(円柱の場合はS_PとS'_Pは一致する)とU、V軸の関係から回転角パラメータθ_1は一意に決まる。

4. このように求めたθ_1を2章で述べた2次曲面同志の相貫曲線の式に代入することにより、第2パラメータa_1および空間点S_Pに対応する相貫曲線上の点C_P(以下、相貫点と呼ぶ)が求まる。このとき、空間点に対応する相貫点が2個存在する場合がある。これは式2.26から分かるように、θ_1に対応するa_1の解が2個存在することに起因する。

5. 相貫点C_Pと中心軸Wを含む平面を設定し、これを被スウィープ形状定義平面X_S-Y_Sとする。この平面は空間点S_P、投影点S'_Pを含み、相貫点が2個存在する場合には、当然、2個とも含む。

前述のように上記の設定方法は空間点S_Pが垂線を下ろす対象2次曲面の中心軸W上にある場合には適用できない。従って、この場合には便宜上、中心軸WとU軸を含む平面をX_S-Y_S平面と規定する。

ところで、空間点S_Pはその位置により対応するθ_1は決まるにもかかわらず、対応する相貫点が存在しないことがある(図3.5)。これは次のことに起因する。すなわち、相貫曲線存在範囲に対応するθ_1の有効域($\theta_{min} \leq \theta_1 \leq \theta_{max}$)は2次曲面同志の結合位置関係から自動的に決まるが(第2章参照)、空間点に対応するθ_1は有効域外の値をも取り得るためである。これは、式2.27において$D<0$の場合に相当する。このような場合には、平面X_S-Y_Sを設定することができないので空間点S_Pから垂線を下ろす対象を他方の2次曲面へ変え、その曲面を利用して相貫点C_Pを求める。ただし、どちらの2次曲面に対してもC_Pが存在しない場合には、本手法は適用できないが、このような場合が発生するのは、与えられた空間点がフィレットから十分に離れている場合と考えられるので、その点に対するペナルティ値を十分な大きさの値にすれば良い。

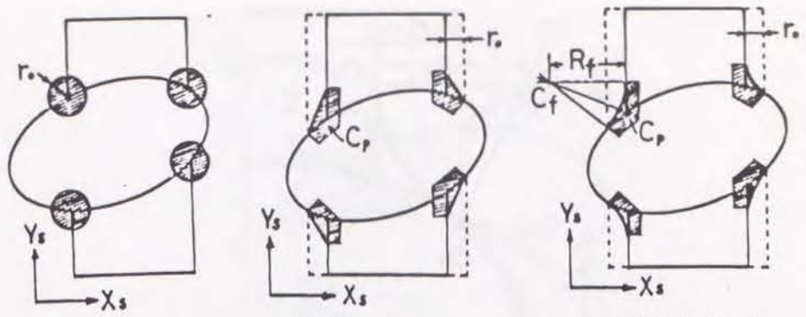


図 3.6: 被スウィープ形状パターン

3.4.3 被スウィープ形状パターンの設定

被スウィープ形状パターンを図3.6に示す。溶接設計におけるのど断面形状に基づき、凸・平・凹形フィレットの3種類を用意した。凸型フィレットは円で代表させることにし半径r_0を入力する。平・凹形フィレットはスウィープ中にその脚長が変化する場合があるので、これを指定し入力するのは困難である。このため、平・凹形フィレットに対しては2次曲面のオフセット値r_0を入力し脚長は計算機内部でその都度計算して求める。

図3.7に示すような凹形フィレットの場合の脚長、凹部用円弧の中心点、半径等は次の手順で求められる。説明の都合上、平面X_S-Y_Sを設定する際に空間点S_Pから垂線を下ろした方の2次曲面をC_1、他方をC_2と記す。C_1とC_2との相貫点C_Pは既に求められているとする。

1. C_1のオフセット曲面とC_2との相貫点O_Sを求め、C_PとO_Sを結ぶ直線分の長さを脚長とする。

2. 相貫点C_Pを中心とし脚長を半径とする円を設定し、C_1との交点O_Rを求める。

3. 点O_Rを通り直線分C_PO_Rに垂直な直線l_R、同様に点O_Sを通り直線分C_PO_Sに垂直な直線l_Sを求める。

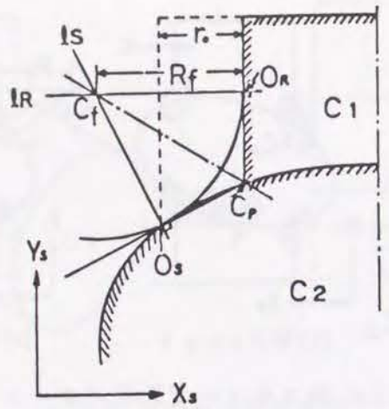


図 3.7: 凹形フィレット用円弧の設定

4. 直線 l_R、l_Sの交点が求める凹形フィレット用円弧の中心点 C_fとなる。

円弧半径 R_fが2点 C_f、O_R(または O_S) 間の距離に等しくなるのは自明である。(ただし、本手法で生成されるフィレットは厳密には、曲面 C_1(または C_2) に接合線において接するとは限らず、直線 C_PO_R(または C_PO_S) と接するだけである。しかし、曲面 C_1と直線 C_PO_R(または C_2と C_PO_S) は十分近いので、フィレットに厳密さが要求されない限り、実用上無視できると考えられる。) また、2点 O_S、O_Rを直線分で結ぶと平形フィレットを作ることができる。

3.5 ペナルティ関数の設定

3次元ソリッド物体形状の表示等を目的とした形状情報作成の方法にペナルティ法[7]がある。これは、物体が定義されている3次元空間内の各座標点に定義物体表面からの距離と関数関係を持つようなペナルティ値を持たせて、逐次、ペナルティ値を判定し物体形状データを作成する方法である。この方法により形状処理を行うためには、対象となる幾何形状要素がペナルティ関数を設定できることが条件である。2次曲面要素を境界として持つプリミティブソリッドはペナルティ関数を容易に設定し得るので、もし、スウィープソリッドのペナルティ関数が設定できるならば、スウィープソリッドは形状処理過程において上記のプリミティブソリッドと同様に扱い得る。したがって、ここではスウィープソリッドのペナルティ関数設定を試みる。

今、図3.8に示すような平面 X_S-Y_Sを想定する。この平面には m(ただし、$m \geq 1$) 個の相貫点があるものとし、それに対応して m 個の被スウィープ形状が設定されてい

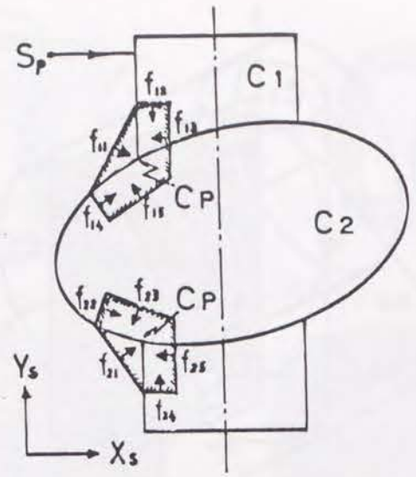


図 3.8: 被スウィープ形状とペナルティ関数

るとする。また、個々の被スウィープ形状はそれぞれ n(ただし、$n \geq 1$) 個の曲線分によって構成されているとする。これらの曲線分を陰関数表現すると平面 X_S-Y_Sは曲線によって二つの領域に分けられる。すなわち、陰関数の値が正となる領域と負となる領域である。ここでは被スウィープ形状の内部向き (図3.8中、斜線部矢印方向) 半空間領域を陰関数値正領域に対応させることにする。このため、被スウィープ形状は有効曲線分の積集合によって表現できるので、これに対応するペナルティ関数 F_iとして次式を与えることができる。

$$\begin{aligned} F_i &= \min(f_{i1}, f_{i2}, \cdots, f_{in}) \\ &= \min_{j=1}^{n}(f_{ij}) \end{aligned} \tag{3.7}$$

ここで、f_{ij}は空間点 S_Pに対する曲線分それぞれの陰関数値である。この平面には m 個の被スウィープ形状が独立して存在するので、被スウィープ形状全体のペナルティ関数 F_Cは、

$$\begin{aligned} F_C &= \max(F_1, F_2, \cdots, F_m) \\ &= \max_{i=1}^{m}(F_i) \\ &= \max_{i=1}^{m} \min_{j=1}^{n}(f_{ij}) \end{aligned} \tag{3.8}$$

となる。式3.8は被スウィープ形状の表面・内部・外部の空間点に対してそれぞれ、ゼ

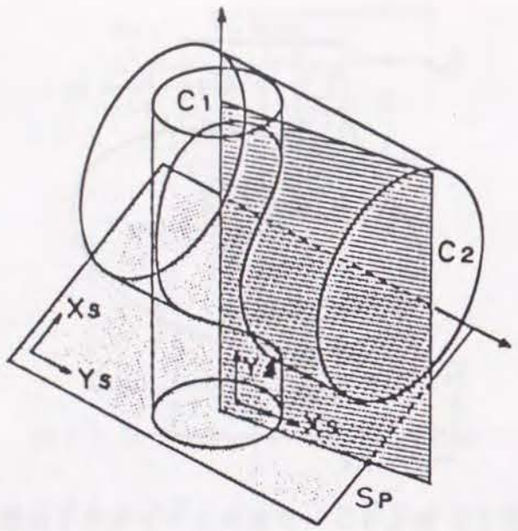


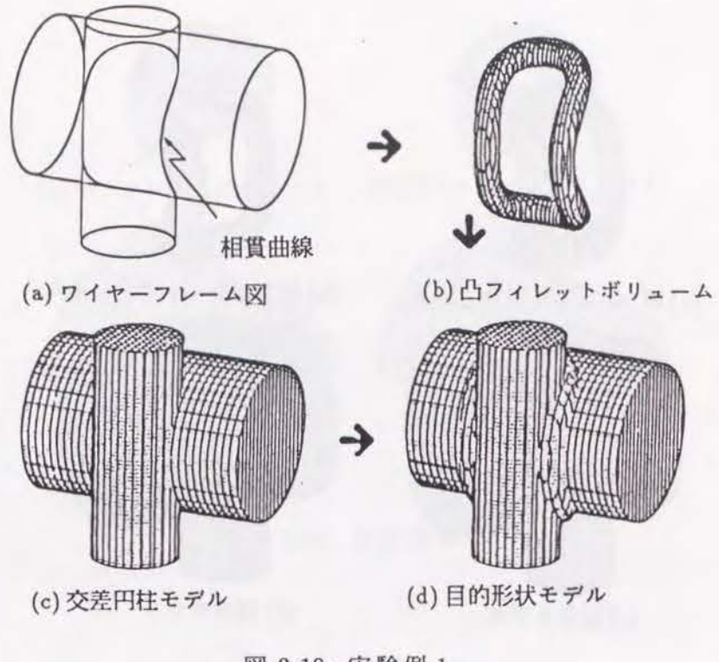
図 3.9: 2種類の $X_S - Y_S$ 平面

ロ・正・負のペナルティ値をもつ。

ところで、図3.9に一例が示されるように空間点 S_P を含む平面 $X_S - Y_S$ が二つ設定できる場合がある。この場合、それぞれの平面において式3.8のペナルティ値は設定されるので、空間点1個に対してペナルティ値は2個求まる。ところが、空間点とペナルティ値は一対一対応を必要とするので次のように定める。説明の都合上、空間点 S_P から2次曲面 C_1 へ垂線を下ろすことによって設定された $X_S - Y_S$ 平面（便宜上、これを C_1 平面と呼ぶ）上で求められたペナルティ値を F_{C1}、同様に、C_2 平面でのペナルティ値を F_{C2} とする。空間点 S_P に対応するペナルティ値 F は、C_1、C_2 平面の有無を判定して次式のように与える。

$$\left.\begin{array}{ll} C_1, C_2\text{平面存在:} & F = \max(F_{C_1}, F_{C_2}) \\ C_1\text{平面のみ存在:} & F = F_{C_1} \\ C_2\text{平面のみ存在:} & F = F_{C_2} \end{array}\right\} \quad (3.9)$$

スウィープの始めから終わりまで平面 $X_S - Y_S$ は任意の空間点に対応づけて設定されるので、この平面上で求めた被スウィープ形状のペナルティ値は、結局、スウィープソリッドのペナルティ値といえる。したがって、式3.9で求まるペナルティ値を逐次判定することによって、スウィープソリッドの形状情報は得られる。



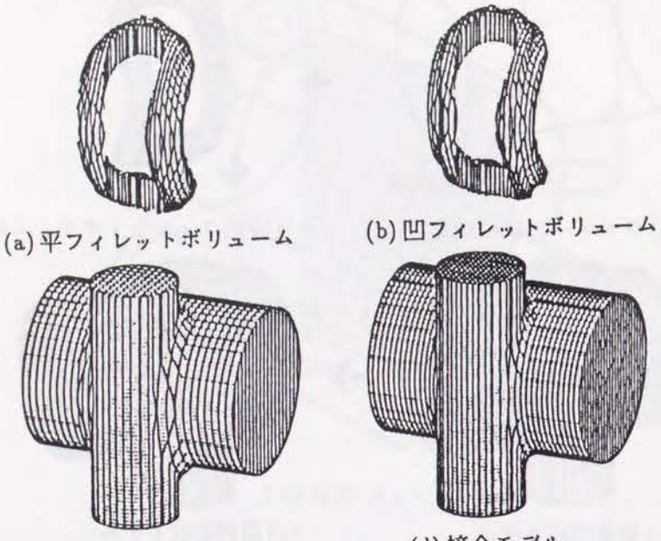
(a) ワイヤーフレーム図　(b) 凸フィレットボリューム

(c) 交差円柱モデル　(d) 目的形状モデル

図 3.10: 実験例 1

3.6 実験結果と考察

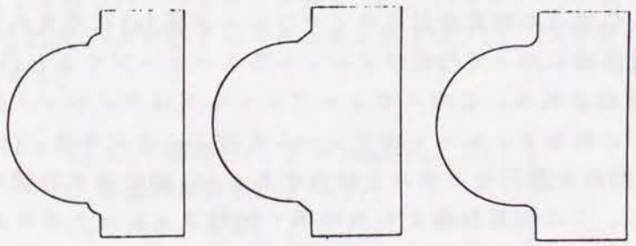
以上で示した手法を汎用 CAD/CAM システム TIPS-1[7] に組み込み、計算機実験を行った。結果の一部を図3.10、3.11、3.12、3.13、3.14に示す。図3.10は円柱同志が交差する例で、この場合の相貫曲線はワイヤフレーム図 (a) に示される3次元空間曲線となる。この相貫曲線に沿って凸形フィレットをスウィープすると (b) 図に示すスウィープソリッドが創成される。このスウィープソリッドはフィレット曲面のソリッドモデルなので、以下これをフィレットボリュームと呼ぶことにする。(b) 図のフィレットボリュームを (c) 図の交差円柱モデルと接合すると (d) 図に示す所望の形状をモデリングできる。同様に、この相貫曲線まわりの平・凹形フィレットボリュームは図3.11(a)、(b) に、また、これを接合したモデルは図3.11(c)、(d) に示される。図3.10、3.11に示したモデルを適当な面で切断した時の断面図は図3.12に示され、凸・平・凹形フィレットがそれぞれ (a)、(b)、(c) 図上で確認できる。図3.13も円柱同志の例であるが、この場合、図3.10の例と比較して結合位置、形状の大きさ等を変えてみた。この図では、被スウィープ形状を構成する曲線分の一部がスウィープ中に変化している様子が



(a) 平フィレットボリューム　(b) 凹フィレットボリューム

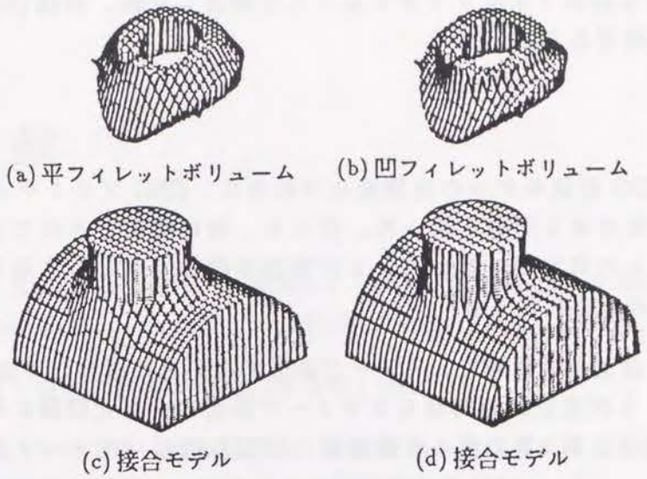
(c) 接合モデル　(d) 接合モデル

図 3.11: 実験例 2



(a) 凸フィレット　(b) 平フィレット　(c) 凹フィレット

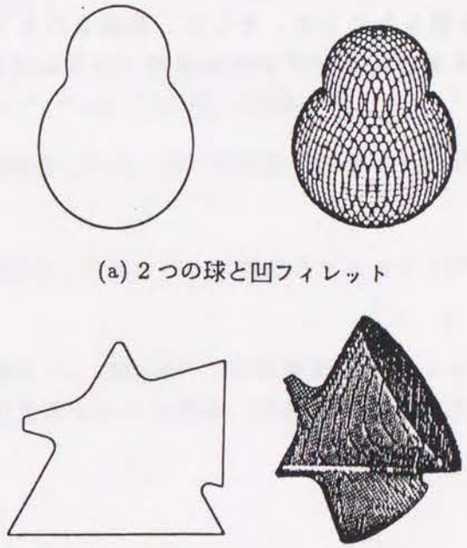
図 3.12: 実験例 3



(a) 平フィレットボリューム　(b) 凹フィレットボリューム

(c) 接合モデル　(d) 接合モデル

図 3.13: 実験例 4



(a) 2 つの球と凹フィレット

(b) 2 つの円錐と凹フィレット

図 3.14: 実験例 5

確認できる。特に凹形フィレットボリュームでは円弧半径が刻々と変化している。図3.14(a)は球同志を凹形フィレットボリュームで接合した例、同図(b)は円錐同志の場合のモデリング例である。

3.7 まとめ

本章では、CSG形状モデルの高精度化のために、CSGプリミティブの接合部分にフィレットを発生させる問題を扱った。中でも、特に困難とされている曲面間の接合部分のフィレットの発生法に対して、2次曲面の場合を例として示した。本章の結果をまとめると次のようになる。

1. フィレット曲面創成問題をスウィープ曲面創成問題に置換して扱う方法を示した。この場合、3次元空間の問題をスウィープ断面の2次元問題に帰着させて扱う点に特徴があり、第2章で求めた曲面間の相貫曲線のパラメータ表現が有効に利用できた。

2. スウィープソリッド用ペナルティ関数の設定方法を示した。

3. いくつかの具体例によって計算機実験を行い、フィレット曲面のソリッドモデルが創成できることを明らかにした。そして、創成されるフィレットボリュームはCSG表現法におけるプリミティブソリッドの一つとして取扱得ることを示した。

参考文献

[1] A. A. G. Requicha and H. B. Voelker : "Solid Modeling—A Historical Summary and Contemporary Assessment", IEEE CG & A, **2**, 2, (1982) 9.

[2] 沖野教郎：自動設計の方法論、養賢堂 (1982) 88.

[3] 長島　忍、木村文彦、佐田登志夫：自由曲面の合成による機械形状の設計、昭和56年度精機学会春期大会学術講演会講演論文集 (1981) 133.

[4] C. Hoffmann and J. Hopcroft : "Automatic Surface Generation in Computer Aided Design", Techinical Report No. TR85-6611, Dept. of Computer Science, Cornell University, (1985) 1.

[5] Y. Shiroma, N. Okino and Y. Kakazu : "Research on 3-D Geometric Modeling by Sweep Primitives", Proc. CAD82, (1982) 671.

[6] 沖野教郎、嘉数侑昇、久保　洋：自動設計プロセサTIPS-1の開発、精密機械、**44**, 3 (1978) 371.

[7] 沖野教郎、嘉数侑昇、久保　洋：自動設計プロセサTIPS-1の設計、精密機械、**42**, 10(1976) 17.

[8] 城間祥之、渡部広一、嘉数侑昇、沖野教郎：フィレットCSG表現法に関する研究—二次曲面相貫線まわりの場合、精密工学会誌、**53**, 2(1987) 308.

第 4 章

CSG から B-Reps への解析的変換

4.1 はじめに

序論で述べたように、CAD/CAM システムの根幹を成す 3 次元形状モデルとしては、CSG（Constructive Solid Geometry）と B-Reps[1]（Boundary Representations）が代表的である。CSG では形状を表現するためのデータ量が小さく、形状定義が容易であるのに対し、B-Reps ではデータ量が大きくデータ構造が複雑であり、形状定義が困難である。また、CSG では形状内外判定は容易である反面、形状表面情報が陽に表現されていないのに対し、B-Reps では形状内外判定は効率が悪くなる反面、形状表面情報は陽に表現されている。このように、それぞれに長所・短所があるため、これら両者の長所を活かす方向として、人間による形状定義は CSG 、内部では CSG と B-Reps をモデルとして持つような二重構造のシステムの実現を図る目的で CSG から B-Reps への自動変換手法の開発が強く要望されている。

この目的のための自動変換手法として、従来は近似的な方法 [2] や探索的な方法 [3] が主に開発されてきているが、変換に要する処理時間や結果としての形状精度の点などで多くの問題を残している。本章では、これらの問題、すなわち、十分な精度を持つ B-Reps モデルを与えられた CSG モデルから高速に発生させる問題を扱う。ここで採用するアプローチ法は、第 2章で述べた曲面間の相貫曲線の解析的厳密解を用いて CSG から B-Reps への変換を可能にするものである。ただし、CSG を構成する面は平面と 2 次曲面（円柱、円錐、球）に限定する。

4.2 CSG と B-Reps

一般に3次元形状Sを表現するためのCSGは式4.1で表すことができる。

$$S = \bigcup_{i=1}^{m} Si = \bigcup_{i=1}^{m} \bigcap_{j=1}^{n} S_{ij} \tag{4.1}$$

ここでS_iはプリミティブと呼ばれ、半空間S_{ij}の積集合として定義される。式4.1を便利に実現するために正、負のプリミティブの和操作のみによる方法と、正のプリミティブのみの和、差操作による方法が存在するが、ここでは前者によるものとする。すなわち、

$$S = (\bigcup_{i=1}^{mP} P_i) \cup (\bigcup_{j=1}^{mQ} Q_j) \tag{4.2}$$

ここで、P_i、Q_jはそれぞれ正、負のプリミティブとする。

式4.1を実現するために開発されたシステムは数多く存在するが、ここでは具体例をTIPS-1[4]システムに限って以下のアルゴリズムを説明する。TIPS-1における式4.1の実現の様相は図1.2(第1章)を参照されたい。TIPS-1においては、各プリミティブS_iはセグメントS_iと呼ばれ、正、負のプリミティブはそれぞれPモードセグメント、Qモードセグメントと呼ばれる。又、各プリミティブはそれを構成する直方体(ドメインD_i)と円柱、球などの半空間(エレメントE_{ij})の積集合として表現される。

一方、B-Repsに関するデータ構造にも種々の方式が提案されているが[2][5][6]、ここでは図4.1に示すような構造を持つB-Repsを採用する。この構造の特徴は、(1)各面ごとのループ、稜線がただちにわかる。(2)データの冗長性が少ない。(3)ある面からループ、稜線を介して他の面へ移動できる(3次元的つながり)などをあげることができる。

このようなCSGからB-Repsへの変換は、図4.2のように行われる。ここで扱うセグメントを構成する曲面は、大きく分けて平面と2次曲面の2種類なので、それらの面データ、すなわち、曲面式はCSGデータより容易に求まる。次に同一面の抽出は、曲面式を比べることにより行われる。同一面の情報は後の有効部分の抽出及び結合関係の作成の時に必要になってくる。以下、相貫曲線、有効部分の抽出、結合関係の作成について述べる。

4.3 相貫曲線

ここで対象としている曲面間の相貫曲線には次の3つの場合がある。(1)平面と平面、(2)平面と2次曲面、(3)2次曲面と2次曲面。(1)の場合は容易に求まる。

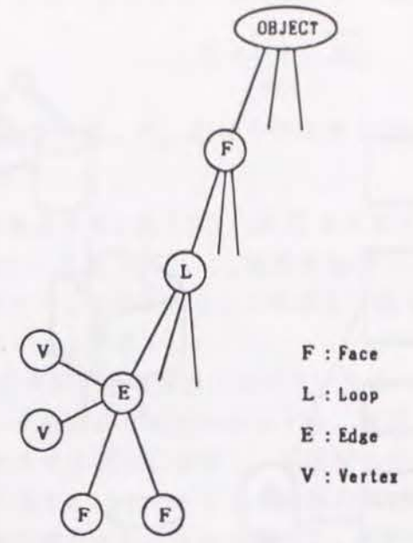


図 4.1: B-Reps のデータ構造

第2章で述べた方法を利用することにより、(2)の場合は空間2次曲線として、(3)の場合はパラメトリック曲線として相貫曲線が得られる。

4.4 有効部分の抽出

第2章で述べた方法により曲面間の相貫曲線は得られるが、この相貫曲線がすべて形状稜線となるわけではない。一般には相貫曲線の一部分が実際の形状稜線となるが、ここではその部分を有効部分と呼ぶことにする。以下に式4.1,4.2で表されるCSG形状に対して有効部分を取り出す方法について述べる。

まず相貫曲線上の1点が有効部分に含まれるための条件を定義する。半空間S_{ij}の境界をB_{ij}、S_iをセグメント、P_i、Q_iをそれぞれ正負のセグメントとすれば、点$\mathbf{X} \in B_{i_1 j_1} \cap B_{i_2 j_2}$が有効部分に含まれるための条件は、次の1、2を満足することである。

1.
$$\mathbf{X} \in S_{i_1} \cap S_{i_2} \tag{4.3}$$

2. S_{i_1}、S_{i_2}がともにQモードセグメントのとき、
$$\mathbf{X} \in \overline{(\bigcup_{i \neq i_1, i_2} Q_i)} \cap (\bigcup_{i=1}^{mP} P_i) \tag{4.4}$$

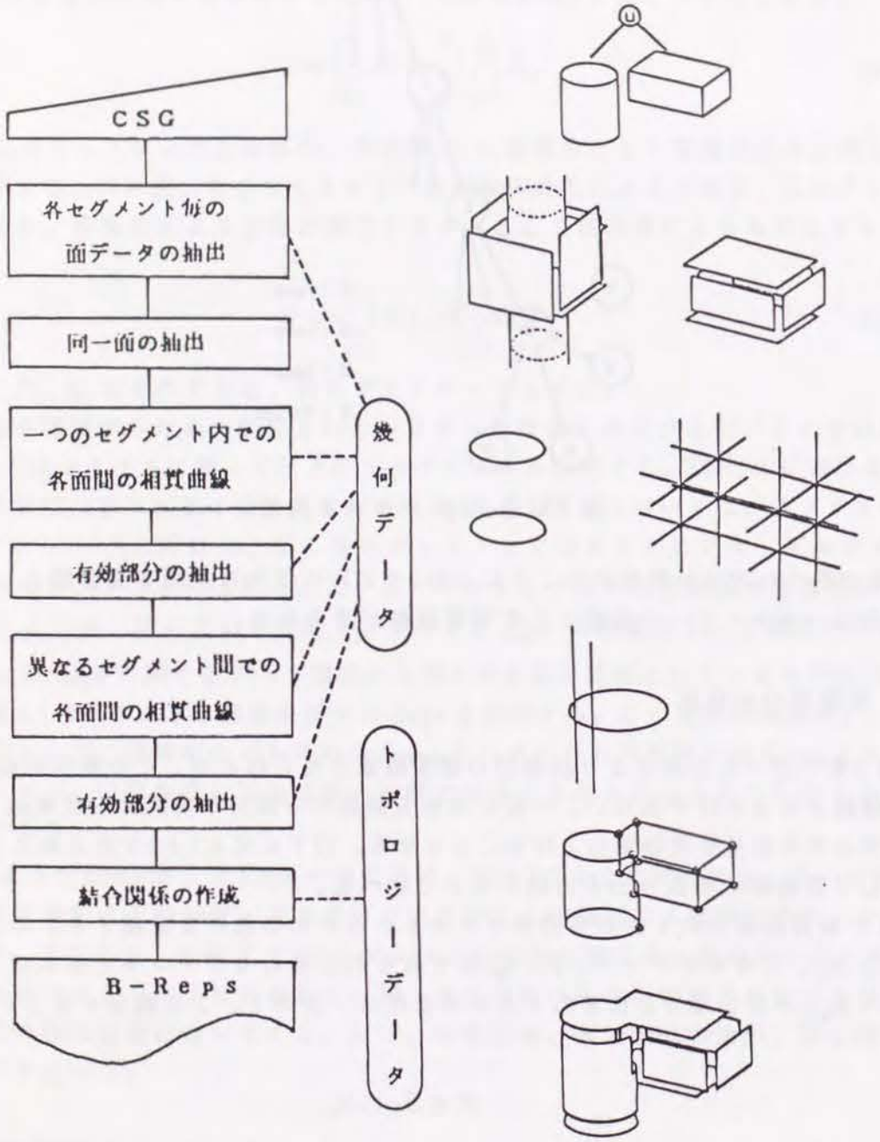


図 4.2: CSG から B-Reps への変換流れ図

S_{i_1}、S_{i_2}の少なくとも一方が Pモードセグメントのとき、

$$\mathbf{X} \in \overline{\left(\bigcup_{i \neq i_1, i_2} S_i \right)} \tag{4.5}$$

ただし、$S_{i_1} = S_{i_2}$なら S_i、P_i、Q_iはその境界を含み、$S_{i_1} \neq S_{i_2}$なら S_i、P_i、Q_iはその境界を含まない。

ここで、各式の意味を補足する。式4.3は、点 $\mathbf{X}$ がセグメント S_{i_1}、S_{i_2}の両方に含まれることを表し、式4.4は、点 $\mathbf{X}$ が S_{i_1}、S_{i_2}以外の Q モードセグメントに含まれず、かつ、ある Pモードセグメントに含まれることを表し、式4.5は、点 $\mathbf{X}$ が S_{i_1}、S_{i_2}以外のセグメントに含まれないことを表す。

次に曲線分としての有効部分の抽出方法について述べる。相貫曲線には、直線、2次曲線、パラメトリック曲線の3種類が存在する。直線、2次曲線の場合は、その線と他の面との交点を求めて小区間に分割し、各区間の中点が条件1、2を満たすかどうかで有効部分を取り出す。パラメトリック曲線の場合は、これを点群曲線として扱い、各点が条件1、2を満たすかどうかを調べて、有効部分を取り出す。

以上から分かるように、曲線分としての有効部分を抽出するために、曲線と曲面との相貫点を求める必要がある。その問題は、直線と平面は1次方程式、直線と2次曲面及び2次曲線と平面は2次方程式、2次曲線と2次曲面は4次方程式を解く問題に帰着させて求めることができる。

しかし例外が存在し、以上のチェックによって有効と判定された曲線分でも、同一面の存在により不要な線分となる場合がある。例えば図4.3において曲線分 $B_{11} \cap B_{21}$ は上記の判定では有効部分となるにもかかわらず、図を見て分かるようにこの曲線分は形状稜線とはならない。これは、面 B_{11} と面 B_{22} とが同一面になるためである。よって注目しているセグメント間に同一面が存在する場合には、上のチェックの他に以下に示す有効性の判定が必要となる。

相貫曲線 $C = B_{i_1 j_1} \cap B_{i_2 j_2}$ に関して、

(a) $B_{i_1 j_1} = B'_{i_2 j_2}$ （面 $B_{i_1 j_1}$ とセグメント S_{i_2} の $B_{i_2 j_2}$ 以外の面 $B'_{i_2 j_2}$ が同一面）

(b) $B'_{i_1 j_1} = B_{i_2 j_2}$ （面 $B_{i_2 j_2}$ とセグメント S_{i_1} の $B_{i_1 j_1}$ 以外の面 $B'_{i_1 j_1}$ が同一面）

(c) $sign(i_1 - i_2) = sign(j_1 - j_2)$

とするとき、(a),(b) どちらか一方を満足するとき、曲線 Cは有効部分とはならない。(a),(b) 両方満足しないとき、又は、(a),(b),(c) すべて満足するとき、曲線 Cは有効部分となる。（条件 (a),(b) 両方満足するときには、同一相貫曲線が2本発生する。条件 (c) は、その内一方のみを有効とするために必要となる。）

$$B_{11} \subset S_1$$
$$B_{21} \subset S_2$$
$$B_{22} \subset S_2$$
$$B_{11} = B_{22}$$

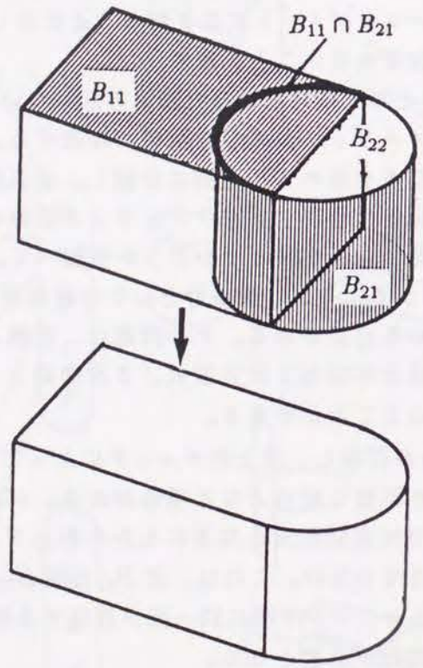


図 4.3: 同一面問題

4.5 結合関係の作成

以上で B-Reps の幾何データが求まるが、これだけでは、面、稜線、点が空間中にばらばらに存在しているだけなので、それが 3 次元のソリッド形状を表しているとはいえない。面、稜線、点の集合が 3 次元ソリッド形状を表現しているためには、それらの間にある決まった結合関係としての位相構造を構成していることが要求される。幾何データを求める段階で、稜線は両端点を持っており、さらに、2 つの面間の相貫線であることがわかっているので、それらの情報を使って結合関係の作成を行う。すなわち、各面に対してその面上に存在する稜線を集め、その両端点の座標値の関係からエッジループが作成され、面、ループ、稜線、頂点の結合関係（図 4.1）が出来上がる。

4.6 システム構築

以上のアルゴリズムを基に、CSG から B-Reps への変換システム EAGLE を構築した（図 4.4）。各モジュールの機能を簡単に述べる。

MINDOM 各セグメント毎のドメイン（存在領域）を最小化する。

SGLIST 各セグメント毎の面データを抽出する。

DMISCK 各セグメントのドメイン同志の干渉チェックを行う。

SAMESF 同一面情報を取り出す。

EDGE 各セグメント内での各面間の相貫曲線を求め、その有効部分を取り出す。

ITST 異なるセグメント間での各面間の相貫曲線を求め、その有効部分を取り出す。

DELSM 同一面による不要線分を消去する。

BREPS 幾何要素間の結合関係を作成する。

view 視線方向を入力する。

SHIL 輪郭線分を生成する。

DRAW 形状の線画表示を行う。隠線処理ができる。

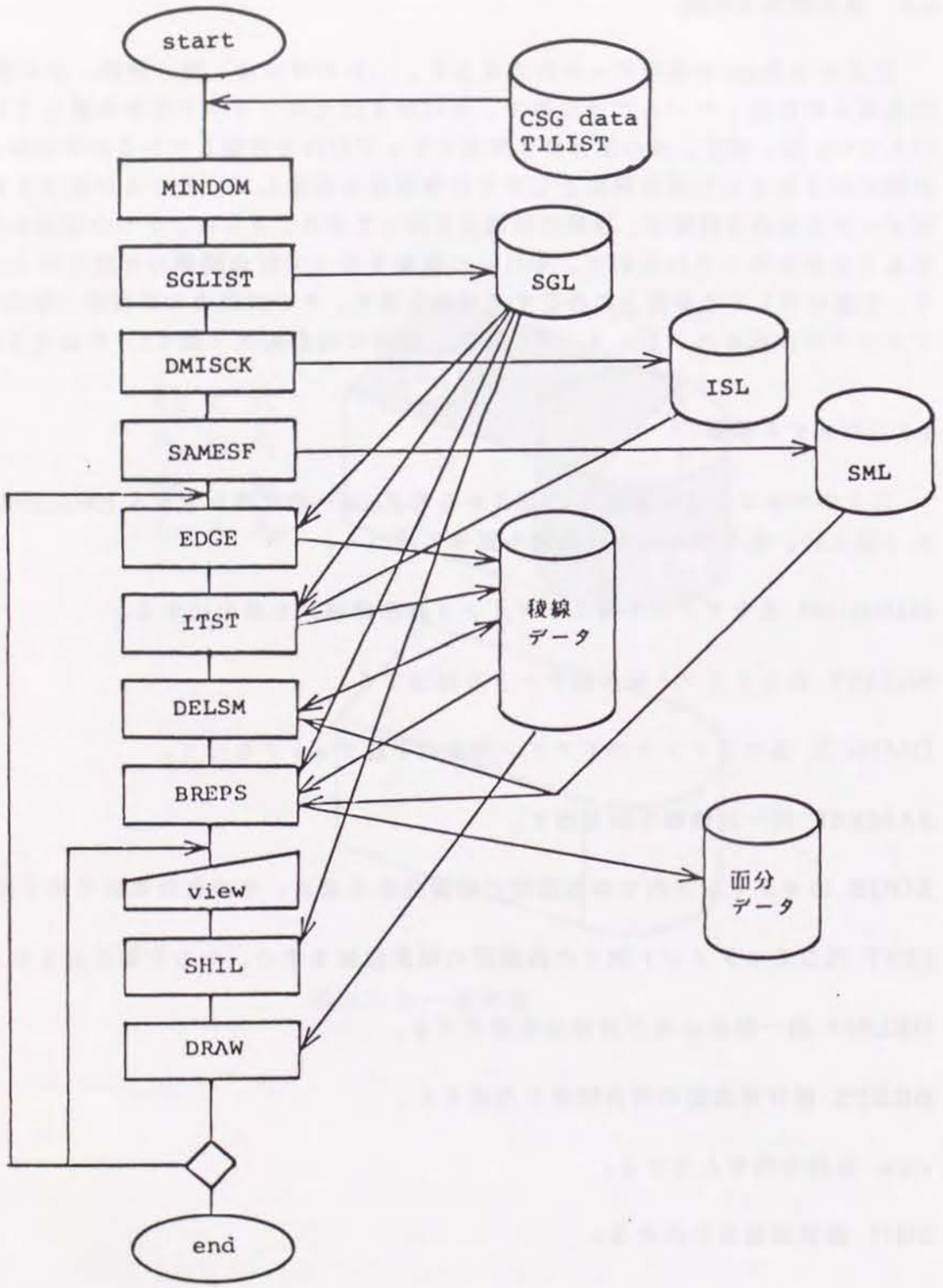


図 4.4: CSG から B-Reps への変換システム EAGLE

EAGLE が出力する B-Reps データを CSG とあわせて CSG/B-Reps 二重構造モデルが生成される。その構造を図 4.5に示す。図中、CoSF は同一面情報を表す。B-Reps の面分に対応するのは CSG のプリミティブの面であるが、同一面が存在する場合は一つの面分に複数のプリミティブが対応する可能性がある。そのために Face から Segment へのポインターは直接ではなく CoSF を間に入れた間接的なものとなっている。また、B-Reps の各幾何要素のデータ構造は以下の 8 種類にまとめられ、それぞれのリスト間のむすびつきは図 4.6のようになる。

フェイスリスト: 形状を構成する各面分に関する情報を持つ（図 4.7）。同一面リスト（LTCSF、LTNSS）を介して CSG と結び付いている。

ループリスト: 面分を構成する各ループに関する情報を持つ（図 4.8）。

エッジリスト: 形状を構成する各稜線に対する情報を持つ（図 4.8）。

平面リスト: 平面の幾何情報をもつ（図 4.9）。

2 次曲面リスト: 2 次曲面の幾何情報を持つ（図 4.9）。

頂点リスト: 頂点の座標値を持つ（図 4.10）。

2 次曲線リスト: 2 次曲線の幾何情報を持つ（図 4.10）。

パラメトリック曲線リスト: パラメトリック曲線の幾何情報を持つ（図 4.11）。

4.7 実験例と考察

ここではいくつかの形状についての変換例を示す。図 4.12(a) は円柱と直方体の和集合として定義された形状で、円柱と直方体の上端面が同一面となっているので、それにより消去されるべき部分は消えている。また、円柱と平面との相貫線（円）の一部が形状稜線となっている様子も分かる。トポロジーデータとして、フェイスリスト、ループリスト、エッジリストを示す。特に上端面については、各リストの丸印の部分が対応している。また、図 4.12(b) は同じプリミティブの積集合として定義された形状で、有効部分（形状稜線）が全く違ってくることが分かる。

図 4.13にやや複雑な形状についての変換例を示す。幾何データはワイヤフレーム図として確認される。トポロジーデータは省略するが、各面分毎に途切れることなくループが構成されることや、浮遊面が存在しないことなどを確認している。また、図中の各

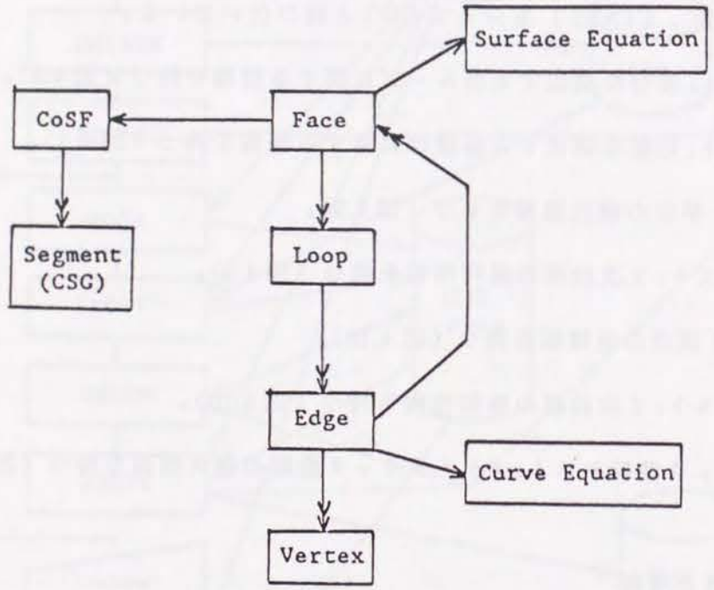


図 4.5: CSG/Breps 二重構造モデル

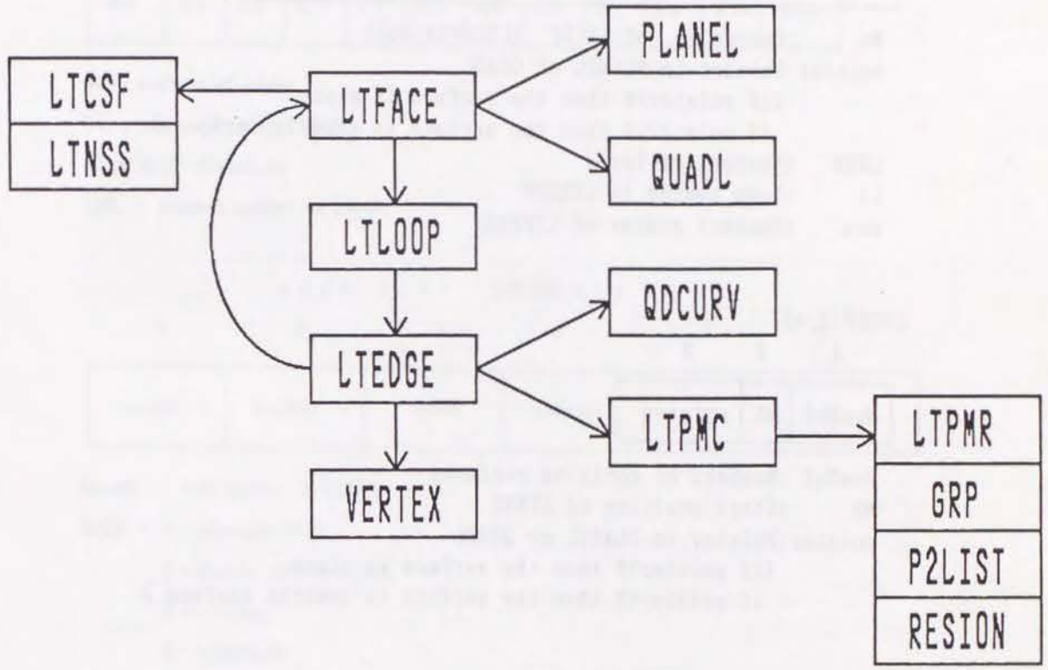


図 4.6: リスト間の結合関係

face list LTFACE(10,*)

1	2	3	4	5	6	7	8	9	10
No	pointer	LNUM	L1	L2	L3	L4	L5	cno.	-

No :Record No. of LTCSF (LTCSF(*,No))
pointer:Pointer to PLANEL or QUADL
(if pointer>0 then the surface is plane,
if pointer<0 then the surface is quadric surface.)
LNUM :Numbers of loops
Li :Loop number in LTLOOP
cno. :Connect number of LTFACE

LTCSF(3,*)

1	2	3
NumSuf	NS	pointer

NumSuf :Numbers of co-lying surfaces
NS :Start position of LTNSS
pointer:Pointer to PLANEL or QUADL
(if pointer>0 then the surface is plane,
if pointer<0 then the surface is quadric surface.)

LTNSS(*)

1
NSS

NSS = 1000*(Segment No.) + (Segment surface No.)

図 4.7: フェイスリスト、同一面リスト

loop list LTLOOP(12,*)

1	2	3	4	5	6	7	8	9	10	11	12
NUM	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	CNO.

NUM : numbers of edges
Ei : edge number in LTEDGE
sign(Ei) : direction
CNO. : connect number in LTLOOP

edge list LTEDGE(6,*)

1	2	3	4	5	6
FaceNO. 1	FaceNO. 2	ICASE	pointer	V1	V2

FaceNO. : face number in LTFACE
ICASE = 1 : straight line
2 : double straight line
3 : ellipse
4 : hyperbola
5 : parabola
99 : parametric curve
pointer : pointer for QDCURV or LTPNC
V1, V2 : pointer for VERTEX

図 4.8: ループリスト、エッジリスト

plane list PLANEL(4, *)

1	2	3	4
a	b	c	d

$ax_1 + bx_2 + cx_3 + d = 0$

quadric surface list QUADL(19, *)

1	2	3	4	5	6	7	8	9	10	11
IPN	a_{11}	a_{22}	a_{33}	a_{12}	a_{23}	a_{31}	a_{14}	a_{24}	a_{34}	a_{44}

12	13	14	15	16	17	18	19
r	r^2	Px	Py	Pz	wx	wy	wz

IPN : kind of quadric surface (IPN = 11 : cylinder 14 : cone 15 : sphere)

a_{ij} : arguments of quadric surface's equation

$$a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + 2a_{12}x_1x_2 + 2a_{23}x_2x_3 + 2a_{31}x_3x_1 + 2a_{14}x_1 + 2a_{24}x_2 + 2a_{34}x_3 + a_{44} = 0$$

r : radius for cylinder or sphere , half of tip angle for cone

(Px, Py, Pz) : point

(wx, wy, wz) : direction vector of axis

図 4.9: 平面リスト、2次曲面リスト

vertex list VERTEX(3, *)

1	2	3
x	y	z

quadric curve list QDCURV(23, *)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ICASE	a	h	b	g	f	c	b1	b2	b3	b4	i	j	k	t1	t2

17	18	19	20	21	22	23
a1	a2	a3	a4	a5	a6	a7

a , h , b , g , f , c : arguments of quadric curve --> $ax_j^2 + 2hx_jx_k + bx_k^2 + 2gx_j + 2fx_k + c = 0$

bi : arguments of plane on which the quadric curve is --> $b_1x_1 + b_2x_2 + b_3x_3 + b_4 = 0$

i , j , k = 1 , 2 or 3 for bi = 0

t1,t2 : start and end point's parameter

ai : constants of quadric curve

図 4.10: 頂点リスト、2次曲線リスト

parametric curve list LTPMC(6, *)

1	2	3	4	5	6
K1	K2	NP21	NP22	NR1	NR2

k1 : pointer at LTPMR for main curve

k2 : pointer at LTPMR for sub curve

NP21 : pointer at P2LIST for main curve

NP22 : pointer at P2LIST for sub curve

NR1 : pointer at RESION for main curve

NR2 : pointer at RESION for sub curve

second parametric curve list LTPMR(31, *)

1	2	3	4	.....
NUM	Nstart	Nend	Nsign	.....

NUM : numbers of parametric segments

Nstart : start position of CRP

Nend : end position of CRP

Nsign : order of parameter (if Nsign > 0 then parameter is increasing
if Nsign < 0 then parameter is decreasing)

group of points list CRP(9, *)

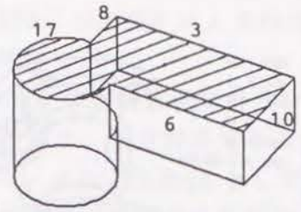
1	2	3	4	5	6	7	8	9
x	y	z	O_1	a_1	$sign_1$	O_2	a_2	$sign_2$

(x,y,z) : 3D coordinate

(O_1, a_1, $sign_1$) : parameters for main curve

(O_2, a_2, $sign_2$) : parameters for sub curve

図 4.11: パラメトリック曲線リスト



(a)

(b)

*** FACE LIST ***
NFACE = 8

SN	NO	POINTR	LNUM	L1	L2	L3	L4	L5	CNO.	-
1	1	6	1	1	0	0	0	0	0	0
2	2	1	1	2	0	0	0	0	0	0
3	3	2	1	3	0	0	0	0	0	0
4	4	3	1	4	0	0	0	0	0	0
5	5	4	1	5	0	0	0	0	0	0
(6)	6	5	1	6	0	0	0	0	0	0
7	7	-1	2	7	8	0	0	0	0	0
8	8	7	1	7	0	0	0	0	0	0

*** LOOP LIST ***
NLOOP = 8

SN	NUM	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	CNO.
1	6	9	14	13	7	-11	-4	0	0	0	0	0
2	4	3	2	-4	-1	0	0	0	0	0	0	0
3	4	-6	15	7	-5	0	0	0	0	0	0	0
4	4	8	16	-9	-1	0	0	0	0	0	0	0
5	4	10	5	-11	-2	0	0	0	0	0	0	0
(6)	5	8	-17	6	-10	-3	0	0	0	0	0	0
7	1	-12	0	0	0	0	0	0	0	0	0	0
8	5	-14	-16	-17	15	-13	0	0	0	0	0	0

** EDGE LIST **

NO	SufNO.1	SufNO.2	ICASE	POINTER	V1	V2
1	2	4	1	0	1	2
2	2	5	1	0	3	4
(3)	2	6	1	0	5	6
4	2	1	1	0	7	8
5	3	5	1	0	9	10
(6)	3	6	1	0	11	12
7	3	1	1	0	13	14
(8)	4	6	1	0	15	16
9	4	1	1	0	17	18
(10)	5	6	1	0	19	20
11	5	1	1	0	21	22
12	7	8	3	1	23	24
13	7	1	3	2	25	26
14	7	1	3	3	27	28
15	3	7	1	0	29	30
16	4	7	1	0	31	32
(17)	6	7	3	4	33	34

(c)

図 4.12: 単純な形状の例

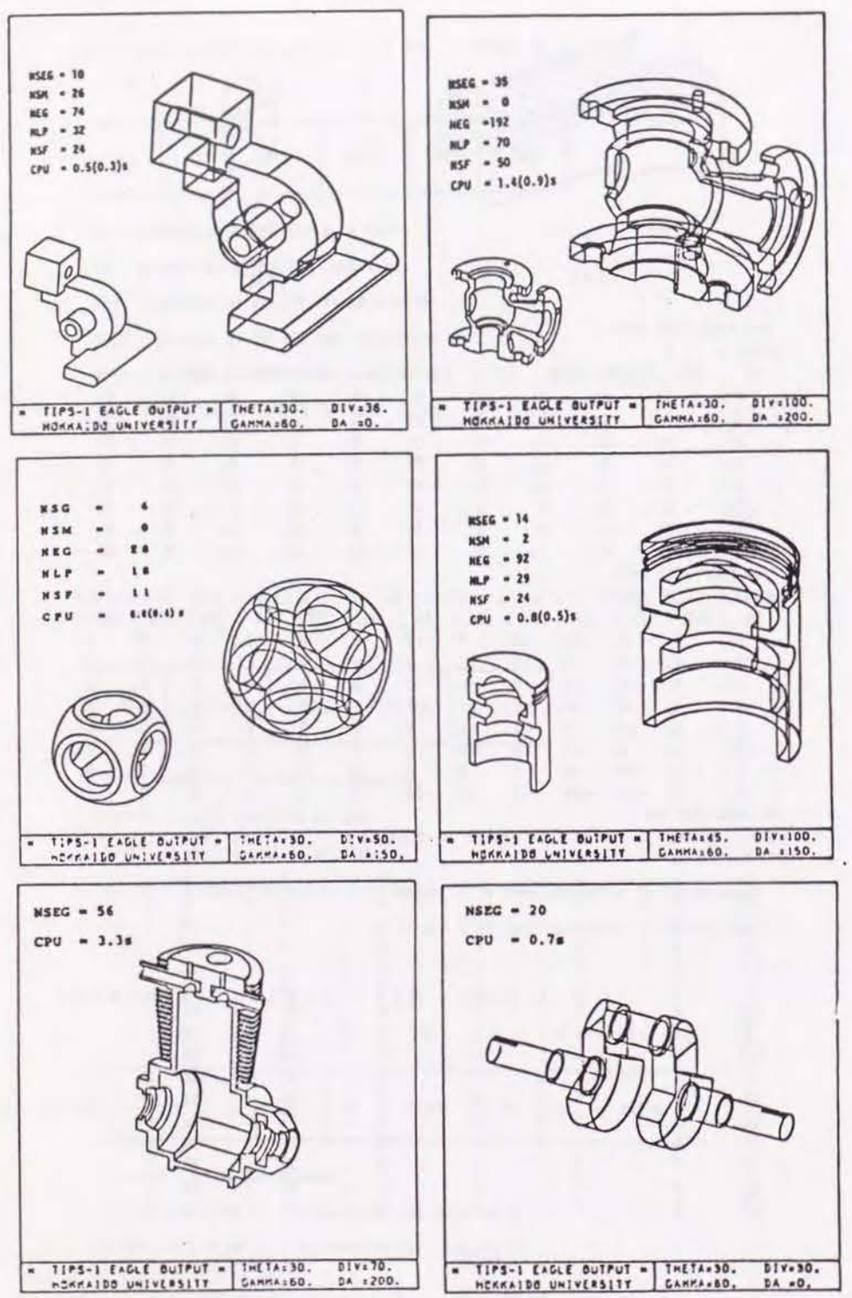


図 4.13: やや複雑な形状の例

値は、NSEG:セグメント数、NSM:同一面数、NEG:稜線の本数、NLP:ループ数、NSF:面分数、CPU:HITAC M-280H による作画まで含めた計算時間、括弧内は作画時間を差し引いた計算時間を示す。

以上の結果から、解析的厳密解を用いて、曲面間の相貫曲線や曲線と曲面との相貫点を求めることにより、十分満足できる速さで、CSG から B-Reps への変換が行われることを確認した。また、B-Reps として得られる面は定義されたままの厳密な曲面であり、稜線はそれらの曲面同志の解析的厳密相貫曲線として求められたものであるから形状精度に関しても十分満足できる結果である。

4.8 まとめ

本章では、形状モデルの高機能化のために、CSG から B-Reps への変換法として、以下の 1 から 4 を特徴とする方法を開発した。すなわち、

1. 平面、円柱、円錐、球面を厳密曲面のまま扱う方法を開発した。

2. B-Reps の形状稜線の候補として、曲面間の相貫曲線を解析的厳密解として求め、さらに、その相貫曲線と他の曲面群との間で相貫点を求めて相貫曲線を部分区間に分割することができた。

3. 相貫曲線上の 1 点の有効判定式、及び同一面の存在に関係する有効性の判定式を与え、形状稜線候補の中から実際の形状稜線を取り出す方法を導いた。

4. 形状稜線の持つ両端点の座標値、及び面に対するポインタ情報から、B-Reps のトポロジーデータを作成し、幾何データと合わせて B-Reps モデルを構築した。

この変換法は、CSG プリミティブを多面体化などの近似を行わずにそのまま B-Reps を生成するので、変換後も形状表現精度は落ちずに、形状モデルの機能が拡張される結果となる。さらに、上記の方法にしたがって CSG から B-Reps への変換システム EAGLE を作成し、変換結果として得られる B-Reps モデルの構造を示し、元の CSG モデルと合わせて CSG/B-Reps 二重構造モデルを構築した。また、いくつかの形状について変換実験を行い、その高速性と高精度性が確認された。

参考文献

[1] "Geometric Modeling Project Boundary File Design(XBF-2)", CAM-I,(1982).

[2] H. B. Voelcker et al. : "Boundary Representation & the BFILE/1 System", PADL System Document, No.10, (1977).

[3] H. Kubo et al. : "TIPS-1 '83 Version, Volume II,III, System Design of Boundary File", Sibley School of Mechanical and Aerospace Engineering, Cornell Univ., (1984).

[4] 沖野教郎、嘉数侑昇、久保　洋：自動設計プロセサTIPS-1の開発、精密機械、44, 3(1978) 371.

[5] B. G. Baumgart : "Geometric Modeling for Computer Vision", Rep. STAN-CS-74-463, Stanford Artificial Intelligence Lab., Stanford Univ., Stanford, Calif., (1974).

[6] I. C. Braid : "CAD Group Document No.101, Notes on a Geometric Modeller", Univ. of Cambridge, Computer Lab., (1979).

[7] 渡部広一、嘉数侑昇、沖野教郎：ソリッド形状モデリングにおけるCSGからB-Repsへの解析的変換の研究、精密工学会誌、**53**、2(1987)315.

第5章

金型CAD用反転形状の自動生成

5.1 はじめに

本章では、前章までに述べた形状モデルの適用例として金型CADを取り扱う。

製品を製造する手段として金型を使う方法が広く利用されているが、この金型の設計は、設計者の経験と勘に依存する部分が大きく、自動化が望まれている。最近では金型設計者のアシスタント的な金型設計用CADシステムが開発されつつあるが、依然として設計者にかかる負担が大きく、抜取り不可能な金型を設計してしまうというような大きな間違いを残したまま金型の設計を終えてしまう危険性もある。この危険性を回避する一つの方法として、マンマシンインターフェースを充実させることにより設計者が間違いを入力する可能性を少なくしたり、正しく設計されているかどうかがグラフィック出力を見ることにより容易に発見できるようにしたりする方法、すなわち、設計者にとってより扱い易いコンピュータシステムを開発していく方向がある。もう一つの手段として、基本的に設計者は何もしなくても済む完全自動化の方法、すなわち、製品形状を入力するだけで金型形状を自動生成してくれる自律的なシステムの構築が考えられる。

本章では、CAD/CAM用シミュレーションプログラムの例として、前者のマンマシンインターフェースを構成するときには、ユーザの定義間違いを指摘するために必要となり、後者のような自律的な金型CADシステムを構築するときには必要不可欠となる、金型製品の抜取り可能性の判定アルゴリズムへの応用を考える。すなわち、製品形状と抜き方向が与えられたときに、実際にその方向に抜取り可能かどうかを判定する方法である。但し、金型は、2プレート（キャビティ、コアのみでサイドコアなどは考えない）とし、形状モデルは第4章で構築したCSG(Constructive Solid Geometry)とB-Reps(Boundary Representations)の二重構造ソリッドモデル[1]とする。

5.2 金型CADシステム

現在一般的な金型CADシステムの概略は図5.1のようなものと思われる。このシステムでは、製品の形状モデルと直方体形状であるブランクモデル、および、抜取り方向（製品形状をブランク形状の中にどの様に位置づけるかという情報）を形状反転プロセサRSGに入力するとRSGはキャビティモデル（図5.2(ｂ)）を生成する。次いで、パーティングラインを入力することにより分割プロセサPPはキャビティ・コアモデル（図5.2(ｃ)）を作成し、金型形状モデルがいったんできあがる。さらに、できあがった金型形状モデルに対して各種の性能解析を行い、解析結果が良好でなければキャビティ・コアモデルに修正を加えたり、もっとさかのぼってパーティングラインの位置を変えたり、抜取り方向を変えたり、ブランクモデルを入力し直したりして最終的には満足できる解析結果が得られるような金型形状モデルが完成する。

ところで、設計者は抜取り方向を入力するときには必ず抜取り可能であるように入力しなければならない。さらに、パーティングラインの指定に対しても同様の注意が必要である。もし誤った指定がされていた場合でも形状反転、分割は正常に行われるので、システムの動きとしては正しく指定された場合とまったく同様にキャビティ・コアモデルが生成されることになる。したがって、このような従来のシステムでは大きな間違いを残したまま金型の設計を終えてしまう危険性があるので、間違いを自動的に判断してくれる機能が必要となる。

5.3 抜取り可能の定義

まず、抜取り可能の定義をする。

定義1 与えられた形状 $OBJ = \{x | f(x) \geq 0\}$[4] が抜取り方向 $\mathbf{v}$ に抜取り可能であるためには以下の条件1から6を満足する H_u と H_l が存在しなければならない（図5.3）。

1. $\mathbf{x} \in H_u \rightarrow \mathbf{v} \cdot \mathbf{N(x)} \geq 0$
2. $\mathbf{x} \in H_l \rightarrow \mathbf{v} \cdot \mathbf{N(x)} \leq 0$
3. $H = H_u \cup H_l$
4. $H_u \cap H_l = \phi^*$
5. $G(H_u, \mathbf{v}) \cap OBJ = H_u$
6. $G(H_l, -\mathbf{v}) \cap OBJ = H_l$

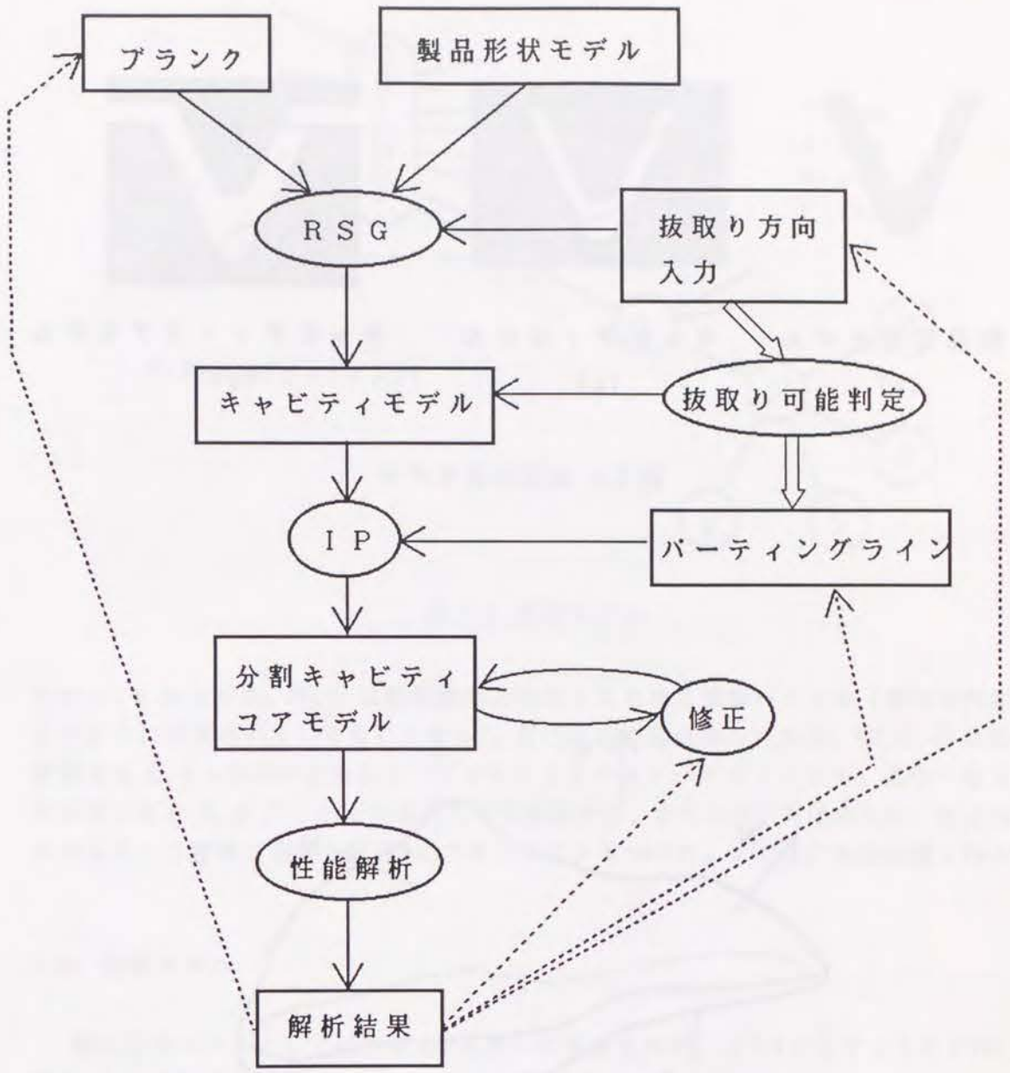


図 5.1: 一般的な金型CADシステム

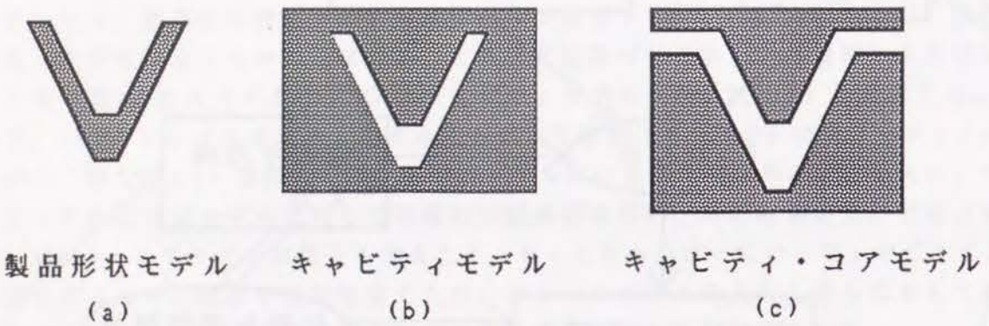


図 5.2: 金型形状モデル

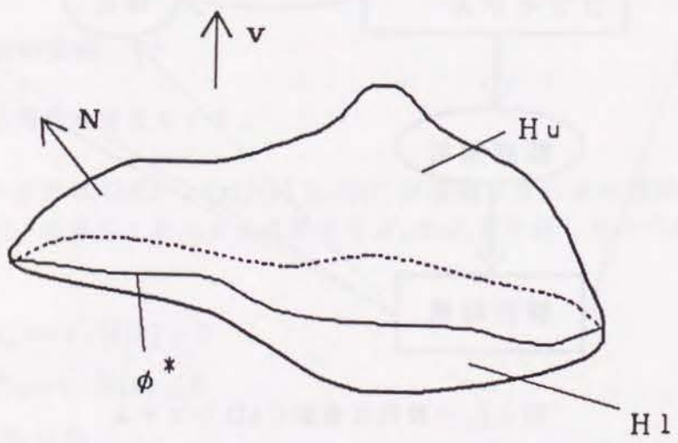


図 5.3: 抜取り可能の定義

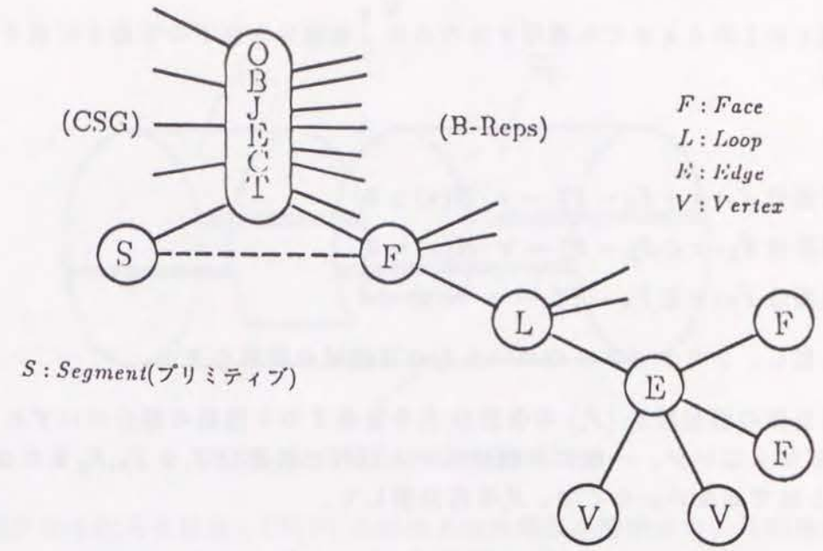


図 5.4: 形状モデル

ただし、x は空間点。N(x) は形状表面上の点 x における法線ベクトル（形状の外側をさすように定義されているものとする）。H は形状表面全体の点集合。$G(H_i, \mathbf{v})$ は形状表面領域 H_i を v方向に十分スイープさせたときのスイープボリュウム。条件 4 は 2 次元領域である H_u と H_l とがその境界のみを共有する、すなわち、共通集合が 1 次元領域になるという意味で空集合記号ϕにアスタリスクをつけた。また、ϕ^*を分割線と呼ぶ。

5.4 形状モデル

製品形状モデルとしては、第4章において構築された、図5.4に示すような CSG と B-Reps からなる二重構造ソリッドモデル [1,2] を仮定している。

5.5 アルゴリズム

第 5.3節の定義 1 にしたがって抜取り可能判定を行うために、第 5.4節で示した形状モデルに施すべきアルゴリズムを以下に述べる。

5.5.1 面分の分類

定義1の1から4までを適用するために、各面分を以下の定義2に示す3種類に分類する。

定義2

正面分 F_P: $\mathrm{x} \in F_P - F_P^* \rightarrow \mathrm{v} \cdot \mathrm{N(x)} > 0$

零面分 F_Z: $\mathrm{x} \in F_Z - F_Z^* \rightarrow \mathrm{v} \cdot \mathrm{N(x)} = 0$

負面分 F_N: $\mathrm{x} \in F_N - F_N^* \rightarrow \mathrm{v} \cdot \mathrm{N(x)} < 0$

ただし、アスタリスクのついたものは領域の境界を表す。

形状全体の面分集合 $\{F_i\}$ の各面分 F_i を定義2の3種類の面分のいずれかに分類しなければならないが、一般に多面体モデル以外は各面分 F_i を F_P,F_Z または F_N に規定することはできない。そこで、F_i を再分割して、

$$F_i = F_{i1} \cup F_{i2} \cup \cdots \cup F_{im}$$

$$F_{ij} \cap F_{ik} = \phi^* \mathrm{or} \phi \quad (j \neq k)$$

とし、各 F_{ij} が F_P、F_Z、F_N のいずれかに分類できるようにする必要がある。この時点では形状全体は面分集合 $\{F_{ij}\}$ として表される。このような F_i の分割及び分類はPascal風に記述したアルゴリズム1のように行える（図5.5）。

アルゴリズム1

```
L ← prof(v, Fi) ;
if L ≠ φ then
  {Fij|j = 1,2,···,n} ← rediv(Fi,L) ;
  for each j
      RP ← repp(Fij) ;
      CF(Fij) ← clss(RP,Fij, v) ;
  end;
else
  RP ← repp(Fi) ;
  CF(Fi) ← clss(RP,Fi, v) ;
end;
```

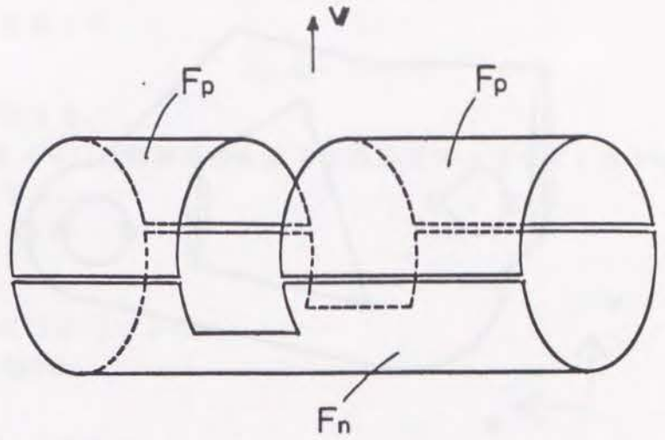



図 5.5: 輪郭線による面分の分割

ただし、RPは3次元空間点、CF(F) は面分 F の分類値が格納され、各関数はそれぞれ以下のような機能を持つものとする。

prof(v,F)..... 面分 F をv方向からみたときの輪郭線を発生させる関数。
rediv(F,L).... 面分 F を曲線Lによって再分割する関数。
repp(F)..... 面分 F 内の代表点を求める関数。
clss(P,F,v)... 面分 F を F_P、F_Z、F_N のいずれかに分類する関数。

また、アルゴリズム1は一般に輪郭線が発生できる面分に対して適用可能である。

5.5.2 分割線分の生成

分割線とは、その上下の面分がそれぞれ定義1における H_u と H_l であるような線分である。したがって、次のものが分割線分となる。ただし、隣接面に正面分と負面分の両方を持つ零面分を極零面分と呼ぶことにする。

1. 正面分と負面分の共有線分
2. 極零面分内（境界を含む）を横切り、抜き方向vに垂直な方向に対して単調連続な線分

1の線分は第5.5.1節で述べた輪郭線分とそれ以外の正面分と負面分の共有線分をさし、これらはすでに生成されているか、あるいは、B-Reps構造から容易に生成できるので、以下2の様な極零面分内における分割線分の生成について考える。

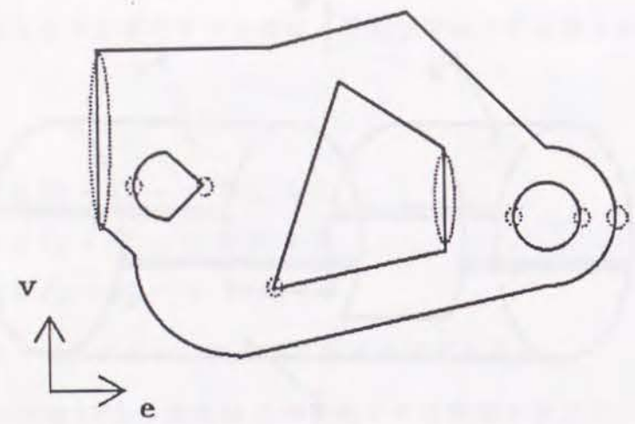


図 5.6: 分割点

極零面分内分割線分の生成可能判定

分割線分の生成可能判定のための準備として、まず、分割点の定義を行う（図5.6）。

定義3 極零面分内において次の2種類のものを分割点と呼ぶ。

1. 隣接面上の分割線分の端点
2. 極零面分同志の共有エッジで正面分と負面分の両方に接続しているもの

さらに、

$$\mathbf{e}\cdot\mathbf{v}=0,\mathbf{e}\cdot\mathbf{N}=0,\mathbf{e}\times\mathbf{v}=\mathbf{N}$$

なる方向eを設定すると分割点は(e,v)パラメータ表現で表すことができるが、分割線分の生成可能性判定のためにはe方向座標だけわかれば十分である。しかも、零面分同志の共有エッジはe方向に垂直となるので2の極零面分同志の共有エッジは点ではないが単一のe方向座標で表現できるので、1の点と同様に扱うことができる。すなわち、分割点はe方向座標のみで表現する。

また、面分はn本のループより構成されているが、各ループ上の分割点の個数は2個を越えてはならない。なぜなら分割点が2個より多いと2つの領域に分割することが不可能となるためである。しかも、ループは閉じているので、分割点の個数は2個以上でなければならない。したがって、2プレートで抜取り可能であるためには、各ループ上の分割点の個数は、ちょうど2個であることが必要条件である。そうすると、

各ループ上の分割点の個数が2個であることを確かめた後、ループi上の分割点ペアをe方向座標を使って、

$$(e_s^i,e_b^i)\quad e_s^i<e_b^i$$

で表すことができる。

以上の考察より、分割線分の生成可能性判定アルゴリズムはPascal風に記述すると次のようになる。

アルゴリズム2

```
if {k|dvpt(k) ≠ 2} ≠ φ then
  end in failure ;
end;
if {i|e_s^i < e_s^1} ≠ φ or
  {i|e_b^1 < e_b^i} ≠ φ or
  {(i,j)|e_s^i < e_s^j < e_b^i, i ≠ 1, j ≠ 1} ≠ φ
then
  end in failure;
end.
```

ここで、dvpt(k)はループk上の分割点の個数を求める関数で、ループ番号1番は外周ループ、その他は内周ループとする。

各極零面分に対してアルゴリズム2を適用して、一つでもend in failureになるものがあれば分割線分生成不可能な極零面分が存在することとなり、抜取り不可能である。

極零面分内分割線分の生成

一般に分割線分の生成には無限通りの可能性があり、その中でどれを選ぶかという問題は製品の出来上り具合いにも影響を与えるので、適当に選んでよいというわけにはいかない。しかし、ここでの目標は抜取り可能判定であるので、抜取り可能の必要条件を満たす限りはどの様な分割線分でもよいこととする。そこで、分割線分生成可能であると判定された極零面分内に次のようにして分割線分を生成する（図5.7）。

1. 内周ループ上の分割点ペアをeの値の小さい順にソートしてループ番号をつけ直す。すなわち、

   $$2\le i<j\rightarrow e_s^i<e_s^j$$

   となるようにする。

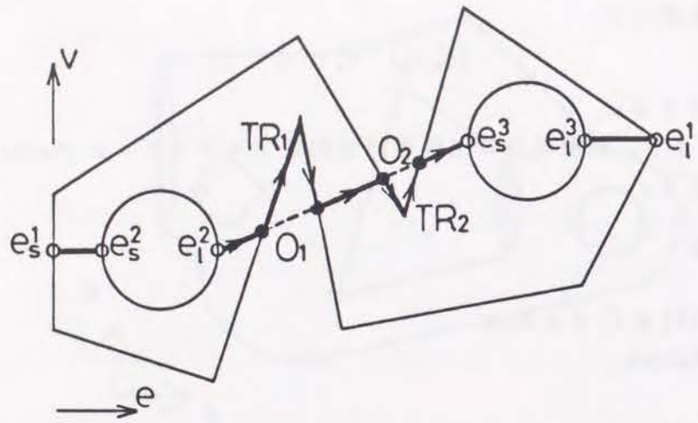


図 5.7: 分割線分の生成

2. e の小さい分割点から順に 2 つずつ結ぶような (e,v) パラメータ空間における直線、すなわち、両端点が以下の分割点の組となるような直線を発生させる。

$$(e_s^1,e_s^2),(e_b^2,e_s^3),(e_b^3,e_s^4),\cdots,(e_b^{n-1},e_s^n),(e_b^n,e_b^1)$$

3. もし、2 で発生させた直線で外周ループと交わり、外周ループの外にはみ出す部分 O_k があれば、O_k の代わりに、その直線が外周ループから切り取る部分 TR_k をつなげて分割線分とする。

5.5.3 分割線分の可視性判定

前節までで、定義 1 の 1 から 4 までを満足する H_u および H_l を生成できたので、次に、それらが定義 1-5,6 を満たすかどうかを調べる。定義 1-6 は 5 と同様なので、以下、定義 1-5 を満たすかどうかを調べる方法について述べる。

定義 1-5 は正領域 H_u を抜取り方向 v に十分平行移動させたときに作るスイープボリュウムと元の形状とが干渉しないということを表しているが、これを次のように言い替えることができる。

P :「H_u 上の全ての点 x は v 方向から見える」。

ただし、命題 P における" 見える" ということを、点 x を始点とし、ベクトル v を方向ベクトルとする半直線と形状 OBJ とが交わらないか、または、接することと定義する（図 5.8)。すると、命題 P は次の命題 P' と同等であることは明かであろう。

P' :「H_u の境界線、すなわち、分割線 ϕ^* 上の全ての点 x は v 方向から見える」。

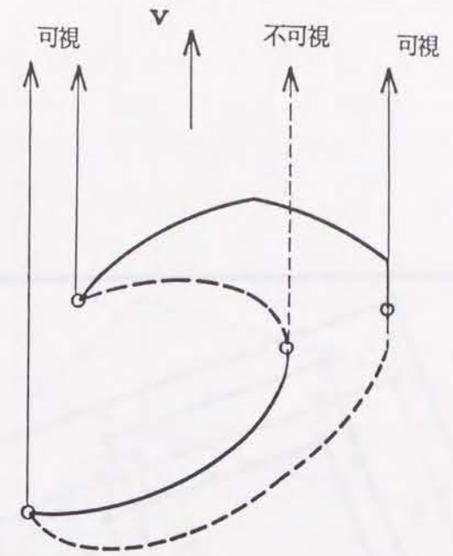


図 5.8: 可視性の定義

したがって、定義 1-5 を満たすかどうかを調べるには、全ての分割線分に対して視線方向を v としたときの可視性を調べればよい。これは、グラフィック出力処理で行うワイヤフレーム図を作成するときの隠線消去法 [3] を用いればよい。すなわち、分割線分の一部でも隠線に含まれれば抜取り不可能であり、全ての分割線分が可視であれば抜取り可能である。

5.6 実験結果

前節のアルゴリズムに基づいてプログラミングを行い計算機実験を行った結果が実験 1 から 3 である。分割線を実線で表示してある。実験 1 は、この図の視線方向を抜取り方向とした場合で、エッジ 21,23 が不可視なために抜取り不可能な例である。実験 2 は、極零面分内に複数個のループがある例で、抜取り方向を紙面上方としたとき全ての分割線が可視となり抜取り可能である。実験 3 も抜取り可能となる例で、上から順に定義形状、分割線、抜取り方向からみた様子である。

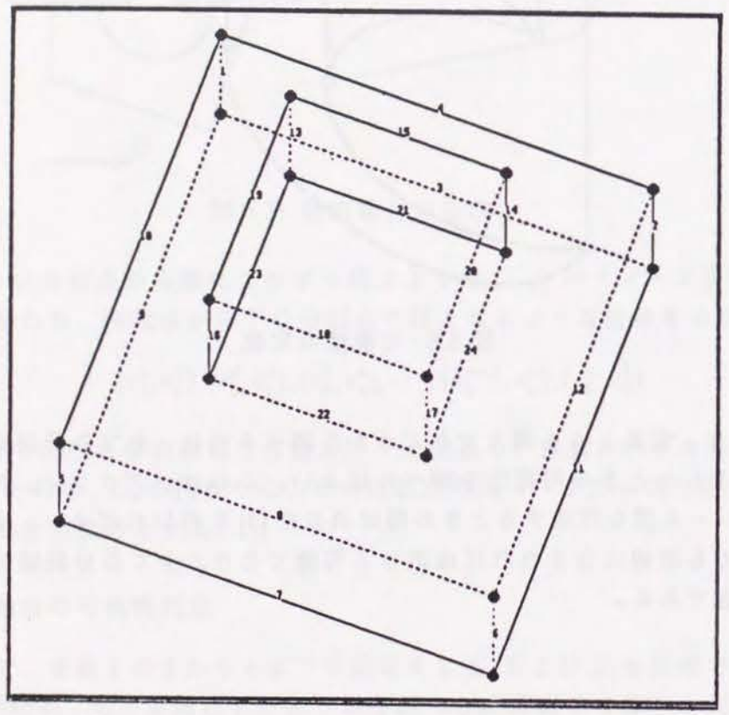


図 5.9: 実験 1

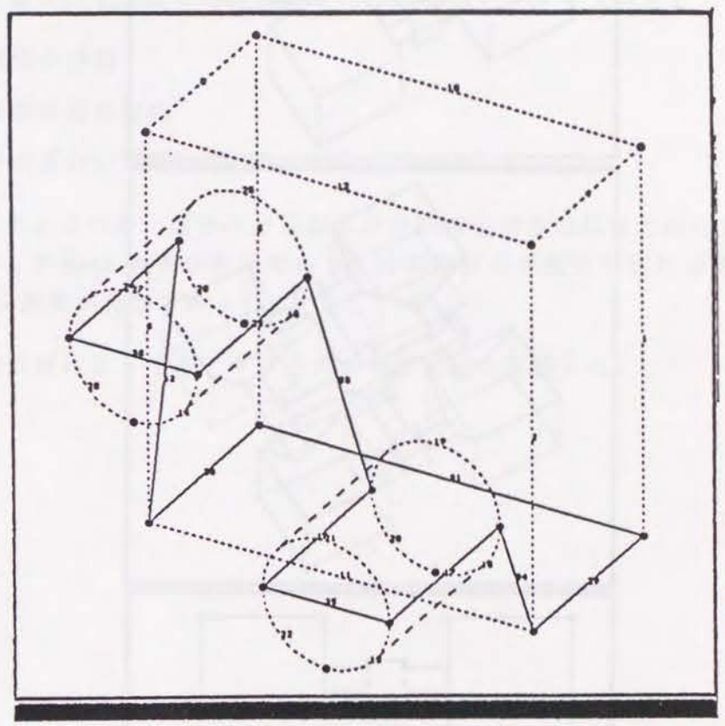


図 5.10: 実験 2

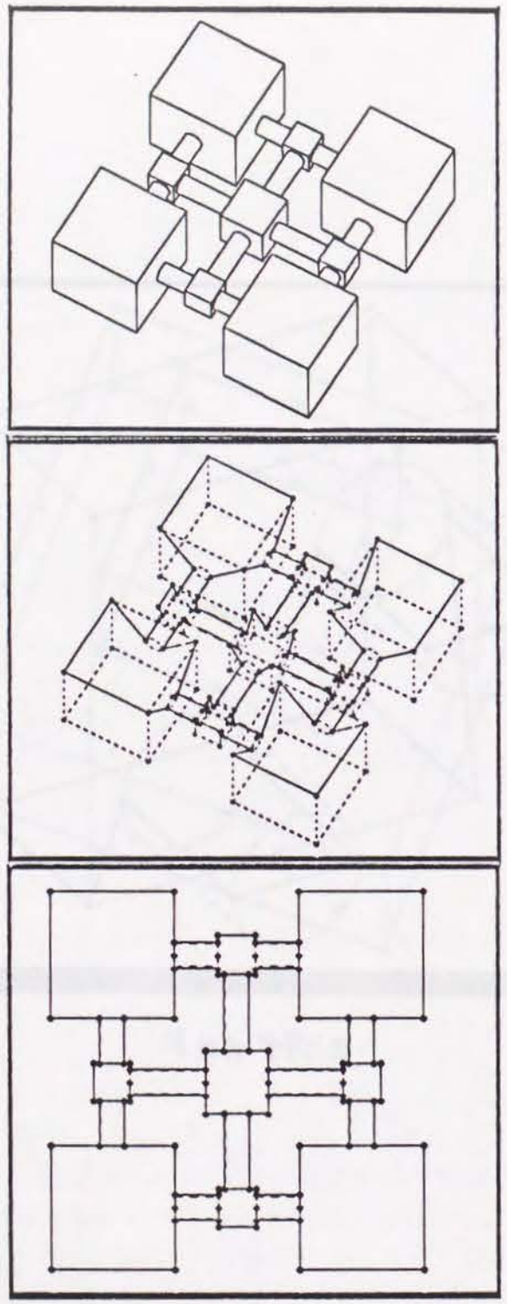


図 5.11: 実験 3

5.7 まとめ

以上本章では、製品形状モデルとしてCSG/B-Reps二重構造モデルを利用し、2プレート金型を用いるという仮定のもとに、抜取り方向が与えられるものとして、その方向に抜取り可能であるかどうかを判定するために、次の1から3のことを行った。

1. 抜取り可能の定義・定式化を行った。
2. 定義に基づいて抜取り可能判定を行うためのアルゴリズムとして、
   - 面分の分類
   - 分割線分の生成
   - 分割線分の可視性判定

   を示した。この際、面分の分類および分割線分の生成には形状の表面情報が必要なので、B-Reps表現が有効であり、分割線分の可視性判定には高速に計算できるCSG表現が有効であった。
3. 計算機実験によって本アルゴリズムの正当性を評価した。

参考文献

[1] 渡部広一、嘉数侑昇、沖野教郎:ソリッド形状モデリングにおけるCSGからB-Repsへの解析的変換の研究、精密工学会誌、**53** 、2(1987)315.

[2] 沖野教郎、嘉数侑昇、久保　洋:自動設計プロセサ TIPS-1 の開発、精密機械、44, 3 (1978)371.

[3] 渡部広一:CSG と B-Reps の二重構造ソリッド・モデルと図形処理、PIXEL No.67 (1988)51.

[4] 渡部広一、嘉数侑昇、沖野教郎:3次元形状モデル間の干渉認識に関する研究、精密工学会誌, **53** No.6 (1987)965.

[5] 渡部広一、沖野教郎、嘉数侑昇:金型CAD用反転形状の自動生成に関する研究—抜取り可能性の判定、精密工学会誌, **56** No.1 (1990)103.

第 6 章
ソリッド形状モデル間の干渉認識

6.1 はじめに

複数の部品より構成される組立製品などに対する形状モデルの構築、あるいは、シミュレーションを行おうとする場合には、各部品の形状モデリングのみならず、各部品形状間の互いの関係のモデリングが重要となる。本章では、この問題を第二の応用例として取り扱う。

計算幾何 (Computational Geometry) において用いる干渉という語は、物体同志が3次元共通領域を持ついわゆるめり込み干渉、2次元的な共通領域を持つ接触干渉に分類することができる。これらを、本章ではめり込み干渉を干渉、接触干渉を接触という語を用いて簡単に表すことにする。実物体では、物体同志が干渉したまま存在することはあり得ないが、計算機内にモデリングされた物体では、いくらでもこの干渉が起こり得る。従って、計算機で形状を扱おうとするとき、実現し得ない状態を排斥するために干渉チェックの機能が不可欠になる。従来、この干渉チェックの方法には、平面近似による方法 [2,3,4] や探索的な方法 [5] が主に用いられているようである。また、台の上に物体が乗っていたり、ロボットが物体を持っていたりする場合、台と物体、ロボットと物体は干渉でなく接触の状態にある。このような状態のときに、干渉しているとか干渉していないとか判断されたのでは不便な場合が多く存在する。それにもかかわらず、現在までこの接触チェックの方法は提案されていないようである。その一つの理由は、形状モデルを多面体などによって近似して扱うためであり、厳密なモデルに対する処理が十分に行われていないというのが現状である。また、接触しているということをチェックできたとしても、単にそれだけでは十分ではない。例えば、台と物体が接触していることが分かったとしても、台の下の面に物体が接触しているだけかもしれず、台の上に物体が安定して乗っているかどうかを調べるためには、ど

のように接触しているかを認識する必要がある。

本章でいう干渉認識とは、計算機内に定義された 2 つの物体が、互いにどのような状態に置かれているのか、すなわち、離れているのか、接触しているのか、干渉しているのかを認識し、そして、接触（干渉）している場合には、接触（干渉）部分を 3 次元形状として解析的に抽出することである。一般に、接触はその 2 次元領域の大きさにより、面接触、線接触、点接触に分類されるが、ここでは面接触のみを扱うことにする。形状モデルは多面体近似などを行わない円柱面と平面からなるものを採用する。また、線分、面分の各状態の分類、定式化を行い、その分類式を用いて干渉認識を行うために、形状モデルに施されるべきアルゴリズムを展開し、さらに、本手法の有効性を確認するために干渉認識システムを構築して実験を行う。

6.2 形状モデル

形状間の干渉認識を行うためには、計算機内に構築される形状モデルはソリッドでなければならない。一般にソリッドモデルには CSG[6] と B-Reps[7] が代表的であるが、形状間の干渉認識がより容易に行えるのはどちらかを考えてみる。CSG では任意の空間点が形状の内側に存在するか外側に存在するかが容易に分かるので、干渉の有無のチェックには比較的適しているようである。しかし、形状の表面状態が陽には表現されていないので、接触状態を認識するのは困難であると思われる。一方、B-Reps では形状の表面状態が陽に表現されているので、接触状態の認識が可能となると思われる。しかし、CSG に比べて空間点の形状内外判定が難しいとされているので、干渉の有無のチェックにはあまり適さないと考えられる。以上の考察から、形状間の干渉認識を行うための形状モデルとしては、CSG/B-Reps の 2 重構造モデルを採用するのが最適であろう。すなわち、最初に CSG が定義されており、そこから CSG に対応づけられた B-Reps が自動生成される形状モデル [1] を採用することにする。この B-Reps モデルは平面近似などはされておらず、2 次曲面と平面からなる厳密モデルである。

6.3 干渉状態と接触状態

2 つの形状が干渉または接触の状態にあるとき、どのような特徴が現れるかを考える。直交座標系の任意点を $X = (x, y, z)$ とすると、形状 A、Bはそれぞれ次式のように表すことができる。

$$A = \{X | f_A(X) \geq 0\} \tag{6.1}$$
$$B = \{X | f_B(X) \geq 0\} \tag{6.2}$$

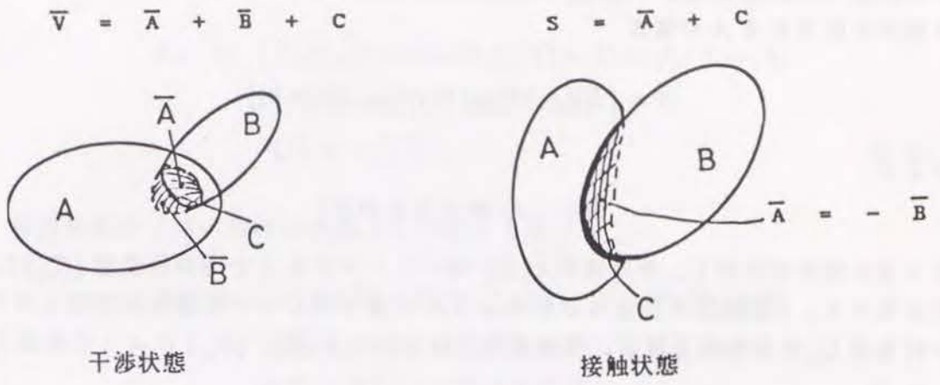


図 6.1: 干渉状態と接触状態の定義

ただし、

$$f_A = F_A(f_A^1, f_A^2, \cdots, f_A^{m_A}) \tag{6.3}$$
$$f_B = F_B(f_B^1, f_B^2, \cdots, f_B^{m_B}) \tag{6.4}$$

で、F_A、F_Bは正規ブーリアン関数（ Regularized Boolean Operations ）[9] である。すると、干渉状態および接触状態はそれぞれ次のように定義できる。（図 6.1）。

$$\text{干渉状態: } A \cap B = V = \{X | f_V(X) \geq 0\} \neq \phi \tag{6.5}$$
$$\text{接触状態: } A \cap B = S = \{X | f_S(X) = 0\} \neq \phi \tag{6.6}$$

一般に、A 、Bは連結だが、V、Sは連結でない。

干渉状態では、(a)$V = A$ 、(b)$V = B$、(c)$V \neq A$ 、$V \neq B$の 3 通りの場合が存在する（ (a) 、(b) の特別な場合として$V = A = B$が考えられる）。(c) の場合には、形状 A の面分 S_iと形状 Bの面分 S_jとの相貫線 C_{ij}が現れる。

$$\begin{aligned} C_{ij} = \{X | & (f_A^i(X) = 0) \cap (f_B^j(X) = 0) \\ & \cap (f_A(X) = 0) \cap (f_B(X) = 0) \cap (f_A^i \neq \pm f_B^j)\} \end{aligned} \tag{6.7}$$

この相貫線を干渉線と呼ぶ。また、Bに含まれる A の境界

$$\bar{A} = \{X | (f_A(X) = 0) \cap (f_B(X) > 0)\} \tag{6.8}$$

A に含まれる Bの境界

$$\bar{B} = \{X | (f_A(X) > 0) \cap (f_B(X) = 0)\} \tag{6.9}$$

が存在する。干渉領域Vは$\{C_{ij}\}$、$\bar{A}$、$\bar{B}$によってバウンドされる。接触状態では、Bの境界に含まれるAの境界

$$\bar{A} = \{X|(f_A(X) = 0) \cap (f_B(X) = 0)\}$$

および

$$\bar{B} = -\bar{A}(\text{面の向きが逆})$$

なるBの境界が存在し、その境界$\bar{A}(\bar{B})$をバウンドするAとBの相貫線$\{C_{ij}\}$(式6.7)が存在する。(接触面が完全な球面のように1面で閉じている場合は存在しない。)この相貫線C_{ij}を接触線と呼ぶ。接触領域Sはこれら$\bar{A}(\bar{B})$、$\{C_{ij}\}$によって構成される。

6.4 線分と面分の状態の分類と定式化

3次元空間内に2つの形状A、B(式6.1、6.2)が置かれているとき、そこには形状の稜線、形状間の相貫線が存在するが、各々の線分は形状間の位置関係によっていろいろな状態になる。一般に線分の状態は着目する面によって変わってくるので、着目面を$S_A^i = \{X|(f_A^i(X) = 0) \cap (f_A(X) = 0)\}$、すなわち、形状$A$の面分$i$としたときの状態定義式を以下に示す。

1. 形状外線分(他の形状の外側に存在する線分)

$$\begin{aligned} E_O \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A^j(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B(X) < 0) \cap (f_A^i \neq \pm f_A^j)\} \end{aligned} \tag{6.10}$$

2. 形状内線分(他の形状の内側に存在する線分)

$$\begin{aligned} E_I \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A^j(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B(X) > 0) \cap (f_A^i \neq \pm f_A^j)\} \end{aligned} \tag{6.11}$$

3. 面接触線分(他の形状の表面に存在する線分で接触面をバウンドする)

$$\begin{aligned} E_S \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A^j(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B^k(X) = 0) \cap (f_B(X) = 0) \\ & \cap (f_A^i = -f_B^k)\} \end{aligned} \tag{6.12}$$

4. 隣面接触線分(他の形状の表面に存在する線分で隣の面分が接触面となる)

$$\begin{aligned} E_N \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A^j(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B^k(X) = 0) \cap (f_B(X) = 0) \\ & \cap (f_A^j = -f_B^k)\} \end{aligned} \tag{6.13}$$

5. 線接触線分(他の形状の稜線上に存在する線分)

$$\begin{aligned} E_L \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A^j(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B^k(X) = 0) \cap (f_B^l(X) = 0) \cap (f_B(X) = 0) \\ & \cap (f_A^i \neq \pm f_A^j) \cap (f_B^k \neq \pm f_B^l)\} \end{aligned} \tag{6.14}$$

6. 干渉線分(他の形状の表面に存在する線分で干渉面をバウンドする)

$$\begin{aligned} E_C \equiv\ & \{X|(f_A^i(X) = 0) \cap (f_A(X) = 0) \\ & \cap (f_B^k(X) = 0) \cap (f_B(X) = 0) \\ & \cap (f_A^i \neq \pm f_B^k)\} \end{aligned} \tag{6.15}$$

一般に、面分は線分の集合によってバウンドされている。次の式は、面分Fが線分$e_1, e_2, \cdots, e_n (n \geq 0)$によってバウンドされていることを表すことにする。

$$F = F(e_1, e_2, \cdots, e_n) \tag{6.16}$$

面分の状態は、他の形状の表面・内側・外側に存在する場合が考えられる。以下に、これらの面分の状態判定式を定義する。

1. 接触面分(他の形状の表面上に存在する面分)

$$F_S \equiv F(e_1, e_2, \cdots, e_n) \tag{6.17}$$

ただし、

$$\begin{gathered} (\forall e_i \in \{E_S\}) \\ \text{かつ} \\ (X \in F \rightarrow X \in \{X|(f_A(X) = 0) \cap (f_B(X) = 0)\}) \end{gathered} \tag{6.18}$$

2. 干渉面分（他の形状の内側に存在する面分）

$$F_C \equiv F(e_1, e_2, \cdots, e_n) \tag{6.19}$$

ただし、

$$(\exists e_i \in \{E_I\})$$

または

$$((\forall e_i \in \{E_S\} \cup \{E_N\} \cup \{E_L\} \cup \{E_C\})$$
$$\text{かつ}(X \in F - \bar{F} \rightarrow X \in \{X | f_B(X) > 0\})) \tag{6.20}$$

3. 形状外面分（他の形状の外側に存在する面分）

$$F_O \equiv F(e_1, e_2, \cdots, e_n) \tag{6.21}$$

ただし、

$$(\exists e_i \in \{E_O\})$$

または

$$((\forall e_i \in \{E_S\} \cup \{E_N\} \cup \{E_L\} \cup \{E_C\})$$
$$\text{かつ}(X \in F - \bar{F} \rightarrow X \in \{X | f_B(X) < 0\})) \tag{6.22}$$

ただし、

$$\bar{F} = e_1 \cup e_2 \cup \cdots \cup e_n \tag{6.23}$$

である。

6.5 システム構築

前節で定式化した分類式を用いて干渉認識を行うための各種アルゴリズムと、それらを統合した干渉認識システムについて以下に述べる。干渉認識システムの構成を図6.2に示す。以下、各ステップについて説明する。

6.5.1 逆向き同一面の抽出

接触面の候補を探すために各面分の曲面式を比較して求める。a_i、b_iを各々の曲面の式の係数とするとき、以下の式を満足する面分を逆向き同一面とする。ただし、式のパターンは平面と2次曲面の2種類あり、各式は正規化されているものとする。

$$a_i + b_i = 0 \quad \text{for all } i$$

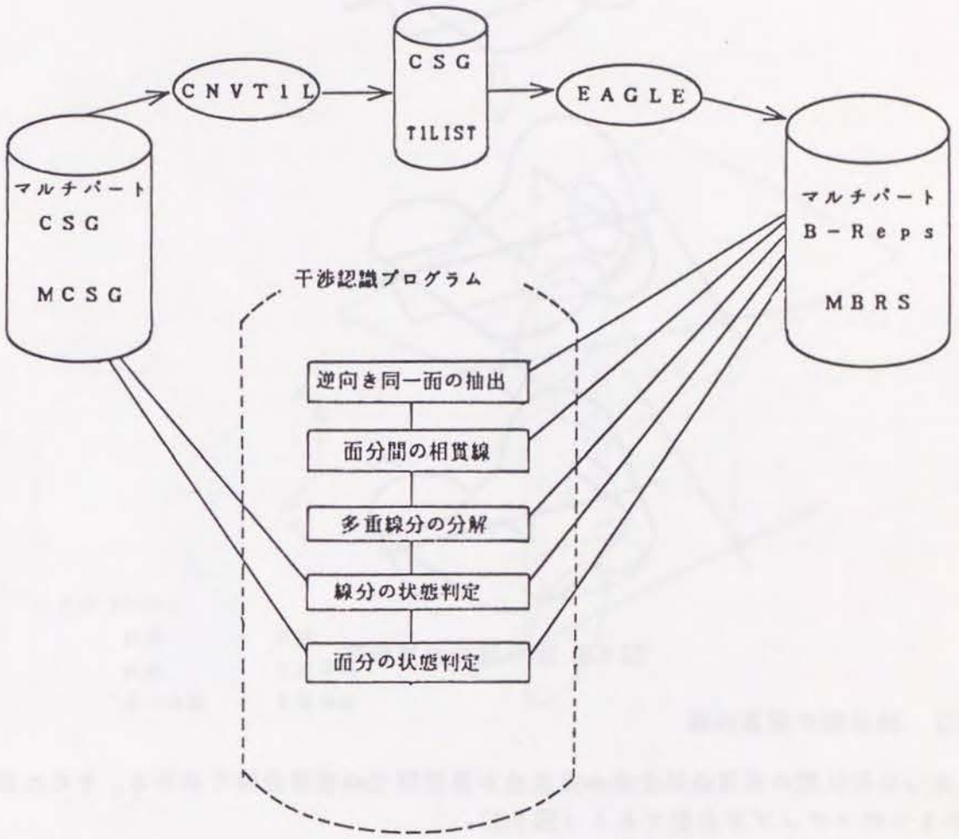


図 6.2: 干渉認識システムの構成

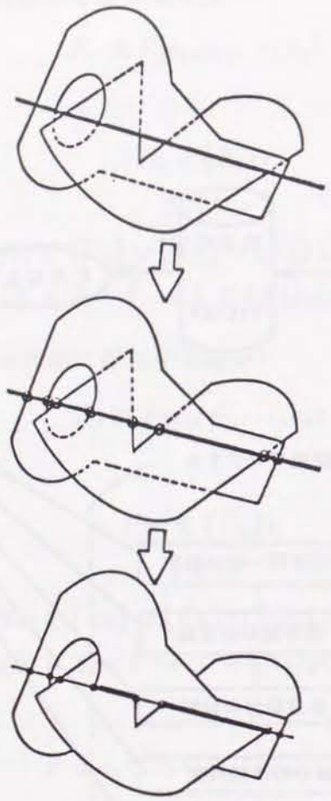


図 6.3: 面分間の相貫曲線

6.5.2 面分間の相貫曲線

2つの形状間の相貫曲線を求めるために各面間での相貫曲線を求める。そのために、次の3つのステップが必要である（図6.3）。

1. 各々の面分が乗っている曲面間の相貫曲線を求める。

2. フェイスループによって相貫曲線を分割する。

3. 有効相貫曲線分（両方のフェイスの内部にある線分）を求める。

ステップ1では、第2章で示した方法 [1] で曲面間の相貫曲線が与えられる。本システムでは曲面を平面と円柱面に限定するが、相貫曲線は直線、2次曲線、パラメトリック曲線の3種類あり、曲面の種類を拡張した場合も同様となる。

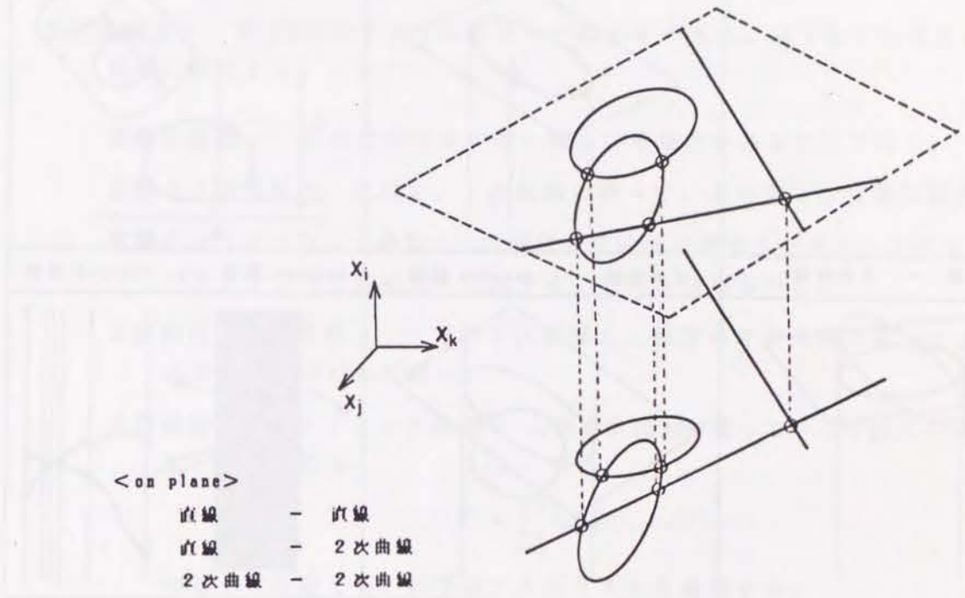


図 6.4: 平面上の曲線間の交点

<on cylinder >

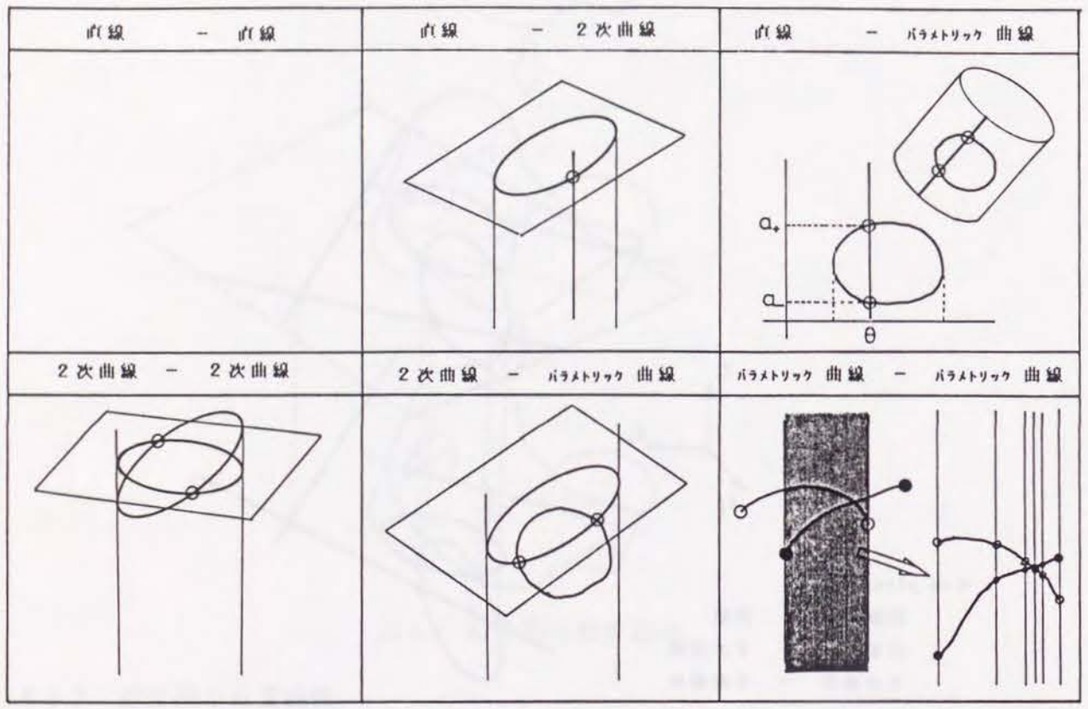


図 6.5: 円柱面上の曲線間の交点

ステップ2では、曲面上において上記曲線間の交点を求めなければならない。平面上での交点、および、円柱面上での交点の求め方は次のようになる。

平面上： 平面式を

$$b_1x_1 + b_2x_2 + b_3x_3 + b_4 = 0$$

とするとき、$b_i \neq 0$ として $j,k \neq i$ なる $x_j - x_k$ 平面に直線、2次曲線を投影して2次元問題として交点 (x_j, x_k) を求め、残りの x_i は上の平面式から得られる。(図 6.4)。

2次曲面上： 図 6.5に示すように6通りの組合せがある。以下各々の場合について簡単に解説する。

直線と直線： 交点は存在せず同一線となる場合があるだけである。

直線と2次曲線： 直線と、2次曲線の乗っている平面との交差問題となる。

直線とパラメトリック曲線： 直線の2次曲面に対するパラメータ値を求めて、パラメトリック曲線式に代入することにより得られる。

2次曲線と2次曲線： 一方の2次曲線と、他方の2次曲線の乗っている平面との交点を求めれば良い。

2次曲線とパラメトリック曲線： b_i を2次曲面が乗っている平面式の係数として、評価関数を

$$f(\theta) = \sum_{i=1}^{4} b_i x_i(\theta),\ x_4(\theta) = 1$$

のように設定して、以下のアルゴリズムを適用する。

Algorithm IP
get θ_1, θ_2 such that $f(\theta_1) \cdot f(\theta_2) < 0$;
while $|\theta_1 - \theta_2| > Acc$ do
  $\theta \leftarrow (\theta_1 + \theta_2)/2$;
  if $f(\theta) \cdot f(\theta_1) < 0$ then
    $\theta_2 \leftarrow \theta$;
  else
    $\theta_1 \leftarrow \theta$;
end;
$\mathbf{x} \leftarrow \mathrm{g}((\theta_1 + \theta_2)/2)$;

ただし、Acc は精度を決める正数、θは第1パラメータ（角度パラメータ）、gはパラメータから空間点の座標を与える関数である。

パラメトリック曲線とパラメトリック曲線： a_iをθに対応する第2パラメータ（長さパラメータ）として、評価関数を

$$f(\theta) = a_2 - a_1$$

のように設定して、アルゴリズムIPを利用する。

ステップ3では、ステップ2で求めた交点によって分割された曲線分の中点を求めて、その点が両方のフェイスの内側にあるか外側にあるかを判定して有効部分を抽出する。フェイス内外判定は判定点を通るフェイス上の直線とフェイスをバウンドするすべての線分との交点の個数を数えることにより行われる。

6.5.3 多重線分の分解

図6.6のように物体O_1が物体O_2上に途中まで乗っている場合を考えてみる。線分ABに注目すると、区間ACは面接触線分で、区間CBは形状外線分となることが期待される。ところが、形状稜線、形状間相貫曲線として求まっているのはAB、ACだけである。ACはABと同一線であるので、CBは、

$$CB = AB - AC$$

として求まる。このように形状稜線と形状間相貫曲線が同一線となって存在しているものを多重線分と呼ぶ。そして、多重線分は同一線部分とそうでない部分とに分割されなければならない。一般的に書くと、

$$E_2 = e_0 - E_1 \tag{6.24}$$

E_2は求めるべき線分の集合、e_0は注目している形状稜線、E_1は$\bar{e_0}$をe_0の一部とするとき、以下の式を満足する線分の集合である。

$$E_1 = \{e_1 | (e_1 \subset O_1 \cap O_2) \cap (e_1 = \bar{e_0})\} \tag{6.25}$$

形状稜線e_0は式6.24,6.25によって分解されて、E_1とE_2になる。

6.5.4 各線分の状態判定

以上の6.5.1節から6.5.3節までの結果を使って、線分の状態分類式1から6によって各線分の状態を判定する。

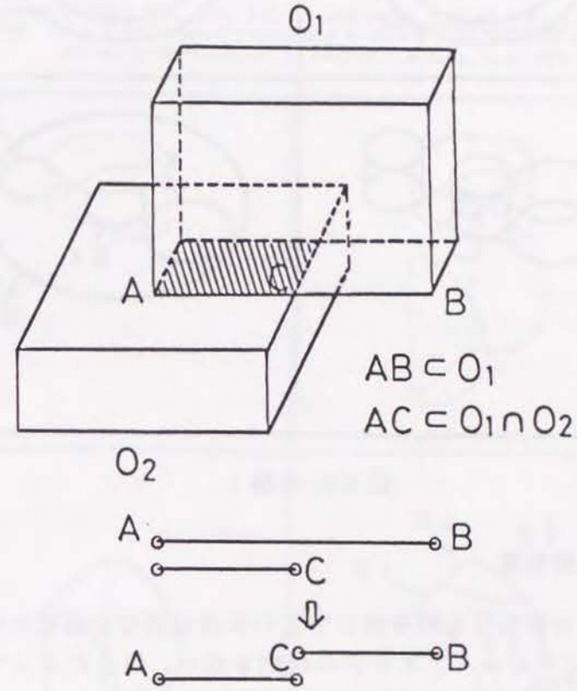


図 6.6: 多重線分

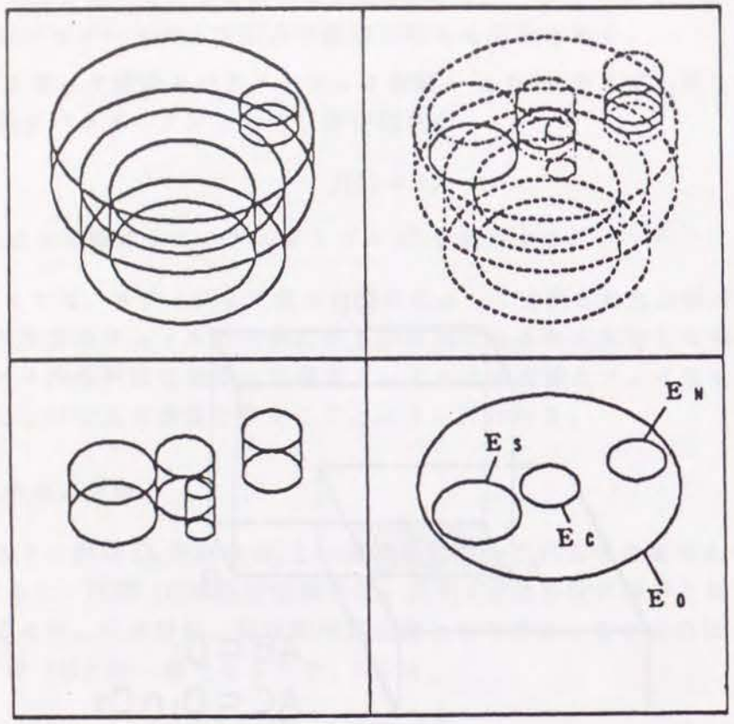


図 6.7: 実験 1

6.5.5 各面分の状態判定

形状の各面毎にその面分を再分割して互いに排他的な小領域に分ける。そして、それぞれの小面分毎にそのループエッジの状態を調べ、場合によっては小面分内代表点を求め、面分状態分類式 1 から 3 を用いて状態判定を行う。

6.6 実験例

前節で述べた干渉認識システムを用いて行った 2 つの実験例を示す。図 6.7では、穴の開いたテーブル（左上）とその上に乗っているように見える 4 つの円柱（左下）が組み合わされた状態（右上）について、各線分の状態判定を行っている。右上図の実線部分は形状間の相貫曲線である。この例では、4 つの円が相貫曲線として現れている。右下図はテーブル上面に関する出力結果で、面接触線分 E_S、干渉線分 E_C、隣面


```
Straight line :
 start point        ( 0.0, 0.0, 5.0 )   end point        ( 4.0, 0.0, 5.0 )
Quadratic curve :
 start point 0.6435( 0.0, 4.0, 3.0 )   end point 1.5708( 0.0, 0.0, 5.0 )
 equation    y²+ z²- 25.0 = 0 .and. x = 0
Parametric curve :
 start point 2.2143( 0.0, 4.0, 3.0 )   end point 3.1416( 4.0, 0.0, 5.0 )
 equation    P = 5.0*( u*cosθ + v*sinθ ) + √(16.0 - 25.0*sin²θ) w
             where, u=(0.0,0.0,-1.0), v=(0.0,1.0,0.0), w=(1.0,0.0,0.0)
```

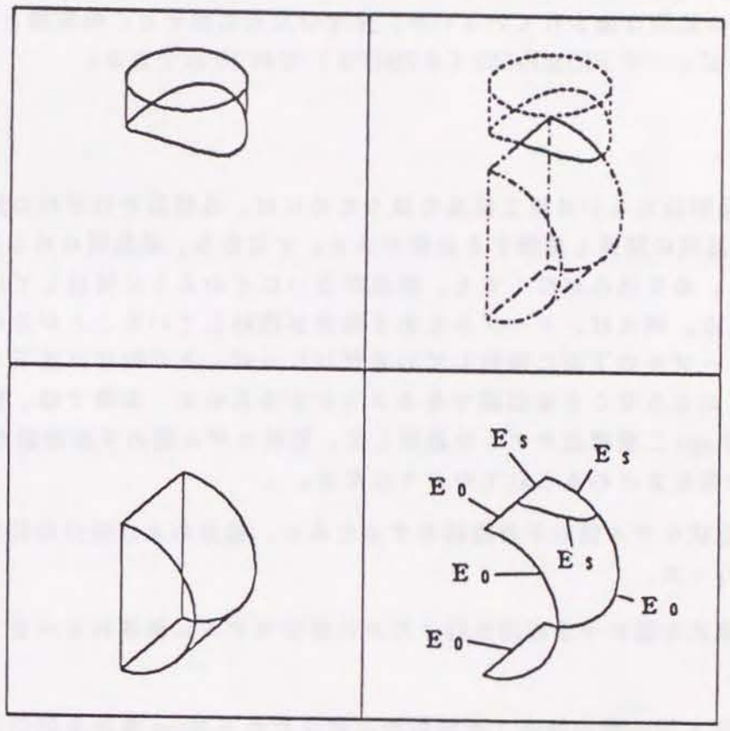


図 6.8: 実験 2

接触線分 E_N、形状外線分 E_O が現れている。最終的に、左側の大きな円柱は下端面全体で接触しており、中間の円柱は干渉しており、右側の円柱は2本の面接触線分で囲まれる円柱面の一部で接触していて、手前の小さな円柱は離れていることがわかる。図6.8は接触面が円柱面で、パラメトリック曲線が絡んでくる場合である。右上図に相貫曲線が実線で表示されている。ちょうど3種類（直線、2次曲線、パラメトリック曲線）の相貫曲線が現れている。図の上部にこれら相貫曲線の数式表現、両端点のパラメータ値および座標値を示す。この例では、これらの相貫曲線がすべて面接触線分となっていて、それらに囲まれる小面分（右下図の上側）は接触面分となっている。

以上の実験は、本方法を用いることにより形状モデル間の干渉認識、特に接触状態の厳密な認識が行えることを確認するために行ったものである。従って、計算時間を短縮するための処理は施されていないが、参考のために示すと、両実験ともCPUタイムはミニコンピュータ PRIME-550（ 0.7MIPS ）で約10秒である。

6.7 まとめ

複数の部品形状あるいは組立製品を扱うためには、各部品それぞれの形状情報のみならず、各部品間の関係も認識する必要がある。すなわち、部品間にめり込みがあってはいけないし、めり込みがなくても、部品が互いにどのように接触しているかを認識する必要がある。例えば、テーブルとある物体が接触していることが分かったとしても、物体がテーブルの下面に接触しているだけならば、その物体は落下してしまうはずである。このようなことを認識できるようにするために、本章では、製品モデルとしてCSG/B-Reps二重構造モデルを適用して、形状モデル間の干渉認識の手法を開発した。その結果をまとめると以下のようになる。

1. 3次元形状モデル間の干渉認識をするために、線分および面分の状態の分類・定式化を行った。
2. 状態分類式を基に干渉認識を行うために形状モデルに施されるべきアルゴリズムとして、
   (a) 逆向き同一面の抽出（面情報が必要なためB-Reps表現を利用した。）
   (b) 面分間の相貫曲線の計算（ B-Reps表現から曲面分の情報を取り出し、第2章の相貫曲線式を用いた。）
   (c) 多重線分の分解
   (d) 線分の状態判定（ CSG表現による形状内外判定法およびB-Reps表現によるトポロジー情報を利用した。）

   (e) 面分の状態判定（面分を囲む線分の状態およびCSG表現による形状内外判定法を用いた。）

   を示した。
3. 3次元形状モデル間の干渉認識システムを構築して実験を行い本手法の有効性を示した。

参考文献

[1] 渡部広一、嘉数侑昇、沖野教郎: ソリッド形状モデリングにおけるCSGからB-Repsへの解析的変換の研究、精密工学会誌、**53** 、2(1987)315.

[2] 重松洋一、嘉数侑昇、沖野教郎 : 3-D 形状モデル間干渉問題の一解法—シンプレックス法による干渉チェック、精密機械、**49** 、11(1983)1561.

[3] J. W. Boyse : "Interference Detection among Solids and Surfaces", Communications of ACM, **22**,1,(1979)3.

[4] K. Maruyama : "A Procedure to Determine Intersections between Polyhedral Objects", Int. J. Comput. and Infor. Sci., **1**,3,(1972)255.

[5] P. G. Comba : "A Procedure for Detecting Intersections of Three-dimensional Objects", J. Assoc. for Computing Machinery, **15**, 3, (1968)354.

[6] 沖野教郎、嘉数侑昇、久保　洋 : 自動設計プロセサ TIPS-1 の開発、精密機械、**44**, 3(1978)371.

[7] "Geometric Modeing Project Boundary File Design(XBF-2)", CAM-I (1982).

[8] Pracleep Shinha : "Surface-Surface Intersection for Geometric Modeling", Ph.D Thesis, Cornell University (1986).

[9] R. B. Tilove : "Set Membership Classification—A Unified Approach to Geometric Intersection Problems", IEEE Trans. Compu., **29**, 10 (1980)874.

[10] 渡部広一、嘉数侑昇、沖野教郎 : 3 次元形状モデル間の干渉認識に関する研究、精密工学会誌, **53** No.6 (1987)965.

第 7 章

組立順序の自動決定

7.1 はじめに

本章は第三の応用として、組立のシミュレーションプログラムの一種で、組立の順序を製品モデルの情報から自動的に生成する問題を取り扱う。組立製品の各部品モデルは、第 4 章で構築した CSG/B-Reps 二重構造モデル [1] を適用し、さらに、第 6 章の干渉認識手法 [3] を各部品モデル間に適用することにより、それらの幾何的接続状態を自動的に取り出す。組立製品における隣の部品同志は接触状態にあるはずであり、第 6 章の干渉認識法では、この接触状態を接触面分として取り出すことができる。各部品は他の部品に接触されることにより、その移動可能方向を制約されている。その制約を干渉認識によって取り出された接触面分の幾何学的形状から推論することにより、各部品が移動可能かどうかを調べ、移動可能ならその可能方向も自動的に抽出することによって、組立順序を自動的に生成することが可能となる。

従来、この組立順序の自動生成問題に対しては、北島 [4]、関口 [5,6]、Laperriere[7] らの研究があるが、これらの方法では、設計者が部品間の接続状態を与えなければならず、設計者にかかる負担が大きい。本手法では、設計者は各部品の形状定義と位置・姿勢定義（ここで定義間違いがあれば干渉認識プログラムによってチェックされる。）を行うだけでよいので、より自動化の進んだシステムの構築に貢献すると考えられる。

7.2 方針

いま、製品モデルが複数の部品よりなり、各々の部品モデルが CSG/B-Reps の二重構造モデルとして与えられ、また、各部品の位置・姿勢が与えられているとする。このような製品モデルに対して組立順序の決定を行うための全体の方針は次のようになる。

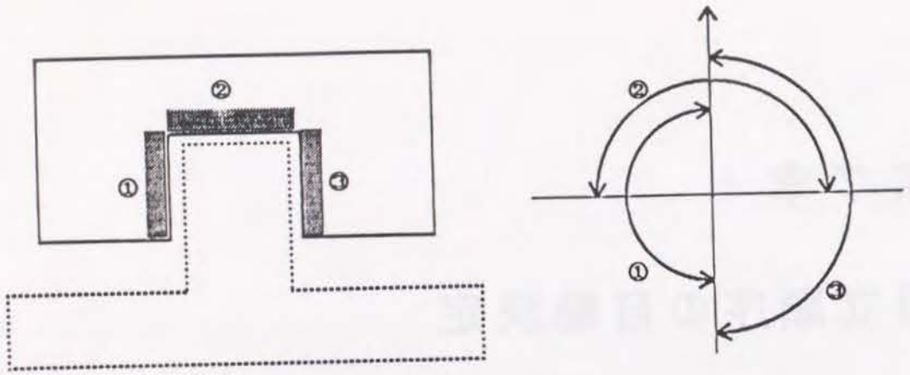


図 7.1: 抜き取り方向の導出

1. 組立順序は分解順序の逆順であると考える。
2. 分解は部品を1個づつ抜き取って行くことにより行う。
3. 抜取りは、抜き取れるものから順に抜き取る。
4. 抜き取れるかどうかの判定に、各部品の接触領域を使う。
5. 接触領域の抽出に干渉認識法を利用する。

5の接触領域の抽出は次のように行われる。与えられた製品モデルのうちの二つの部品モデルを取り出し（部品A、部品Bとする）、第6章で述べた干渉認識システムへ入力すれば、干渉認識システムはそれら2部品間の干渉状態を出力する。すなわち、離れているか、干渉しているか、接触しているかである。もし干渉していれば、これは形状定義に失敗しているので、定義し直す必要がある。離れている場合は、その2つの部品は無関係である。接触している場合は、接触領域も同時に出力されるので、その接触領域を部品A、部品B各々に記憶させる。このような、2者間の干渉認識を全ての組合せについて実行すると、各部品の接触領域が得られる。

以下、抜き取り可能判定法、分解順序決定アルゴリズムについて述べる。

7.3 抜き取り方向の導出による抜き取り可能判定

ある部品Aが他の部品に一つの面で接触しているとき、部品Aはその接触面に沿う方向かその面からはなれる方向に移動可能である。もし、部品Aが他の部品と複数

個の面で接触しているならば、部品Aの移動可能方向は各面に対する部品Aの移動可能方向の積集合として表される（図7.1）。接触面 i に対する部品の移動可能方向 $\mathbf{V}_i$ は次式のように表される。

$$\mathbf{V}_i = \{\mathbf{v} | \mathbf{v} \cdot \mathbf{n}_i(\mathbf{x}) \geq 0\} \tag{7.1}$$

ここで、$\mathbf{n}_i(\mathbf{x})$ は接触面分上の任意の点 $\mathbf{x}$ における単位法線ベクトルである。そうすると、n個の接触面分を持つ部品の移動可能方向は次のように表せる。

$$\mathbf{V} = \bigcap_{i=1}^{n} \mathbf{V}_i \tag{7.2}$$

もし、$\mathbf{V} = \phi$ ならばその部品は抜き取り不可能である。

7.3.1 接触面分 i による移動可能方向 $\mathbf{V}_i$

接触面分が平面の場合は、$\mathbf{n}_i(\mathbf{x})$ が $\mathbf{x}$ に対して不変であるので、$\mathbf{V}_i$ は平面 $\mathbf{n}_i \cdot \mathbf{x} = 0$ を境界とする半空間となる。

接触面分が円柱面の場合は、$\mathbf{n}_i(\mathbf{x})$ は $\mathbf{x}$ に対して一定ではない。ところが、もし円柱面上の点を角度パラメータ θ と長さパラメータ a で表すと、法線ベクトル $\mathbf{n}_i$ は a に対して不変である。すなわち、

$$\mathbf{n}_i(\mathbf{x}) = \mathbf{n}_i(\theta)$$

したがって、接触面が円柱面の場合の移動可能方向は次のように表せる。

$$\mathbf{V}_i = \{\mathbf{v} | \mathbf{v} \cdot \mathbf{n}_i(\theta) \geq 0, \theta_{min} \leq \theta \leq \theta_{max}\} \tag{7.3}$$

接触面分内の最小角度パラメータ θ_{min} および最大角度パラメータ θ_{max} は接触領域の境界上、すなわち、接触線上にある。そして、式7.3が式7.4と同等であることは明かであろう。

$$\mathbf{V}_i = \{\mathbf{v} | \mathbf{v} \cdot \mathbf{n}_i(\theta_{min}) \geq 0\} \cap \{\mathbf{v} | \mathbf{v} \cdot \mathbf{n}_i(\theta_{center}) \geq 0\} \cap \{\mathbf{v} | \mathbf{v} \cdot \mathbf{n}_i(\theta_{max}) \geq 0\} \tag{7.4}$$

ここで、$\theta_{center} = (\theta_{min} + \theta_{max})/2$ である。式7.4は、円柱接触面分に対する移動可能方向は、平面を境界とする半空間3つの積集合として表せることを示している（図7.2）。

7.3.2 移動可能方向 $\mathbf{V}_i$ の積集合の導出

式7.2で表される部品の移動可能方向 $\mathbf{V}$ を各接触面分に関する移動可能方向の集合 $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n\}$ から導くことを考える。$n = 1$ の場合、明かに $\mathbf{V} = \mathbf{V}_1$ である。$n = 2$

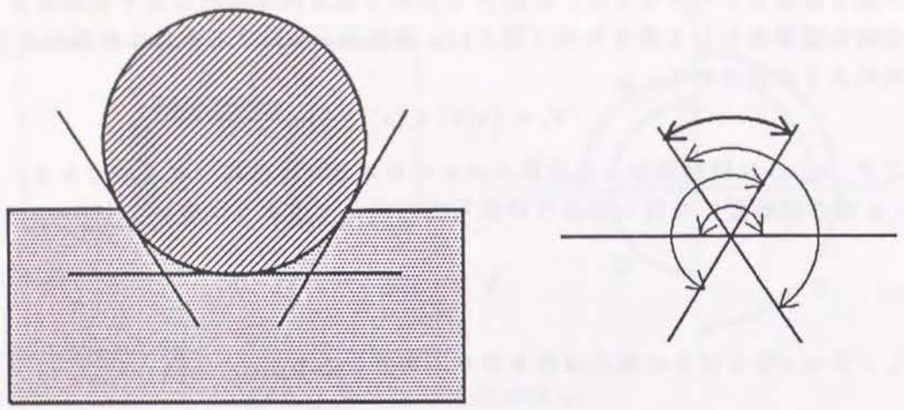


図 7.2: 円柱接触面分による移動可能方向

の場合、Vは式7.5で表される。

$$\mathbf{V} = \begin{cases} \{\mathbf{v}|\mathbf{v}\cdot\mathbf{n}_1 = 0\} & \text{if } \mathbf{n}_1\cdot\mathbf{n}_2 = -1 \\ \mathbf{V}_1 \cap \mathbf{V}_2 & \text{otherwise} \end{cases} \tag{7.5}$$

$n \geq 3$ の場合、帰納法を適用する。記号 S_k が $\mathbf{V}_i$ の最初の k 個の積集合を表すとする。すなわち、

$$S_k = \bigcap_{i=1}^{k} \mathbf{V}_i \tag{7.6}$$

さらに、d は直線の単位方向ベクトルを表し、p（または $\mathbf{p}_s, \mathbf{p}_{s_1}, \mathbf{p}_{s_2}$）は平面の単位法線ベクトルを表すものとすると、$S_k$ は次の 7 種類の場合に分類できる（図7.3）。

1. 空集合：

$$S_k = \phi$$

2. 半直線：

$$S_k = \{\mathbf{v}|\mathbf{v} = t\mathbf{d}, t \leq 0 \ or \ 0 \leq t\}$$

3. 全直線：

$$S_k = \{\mathbf{v}|\mathbf{v} = t\mathbf{d}, -\infty < t < \infty\}$$

4. 全平面：

$$S_k = \{\mathbf{v}|\mathbf{v}\cdot\mathbf{p} = 0\}$$

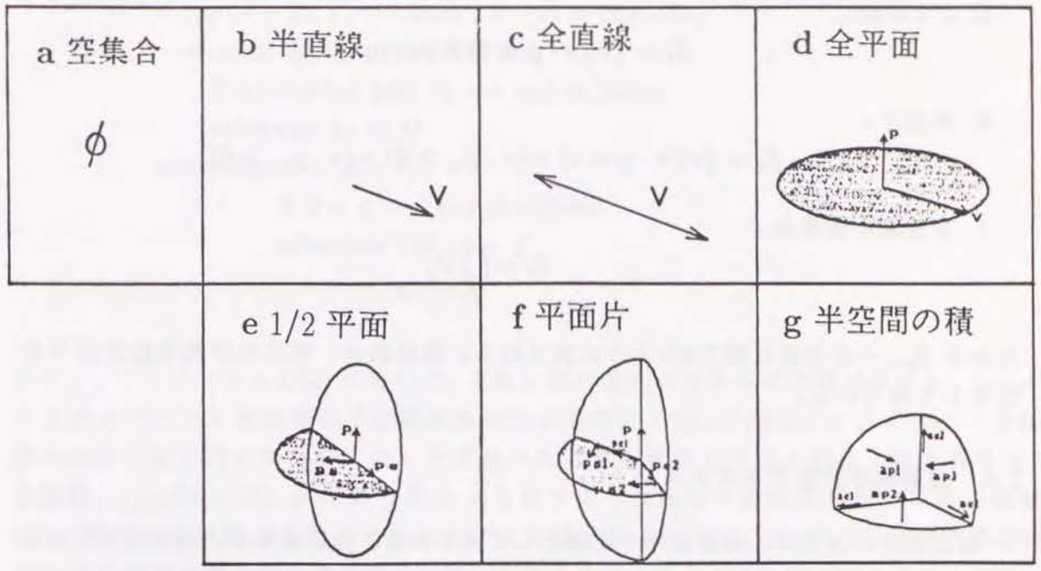


図 7.3: S_k の分類

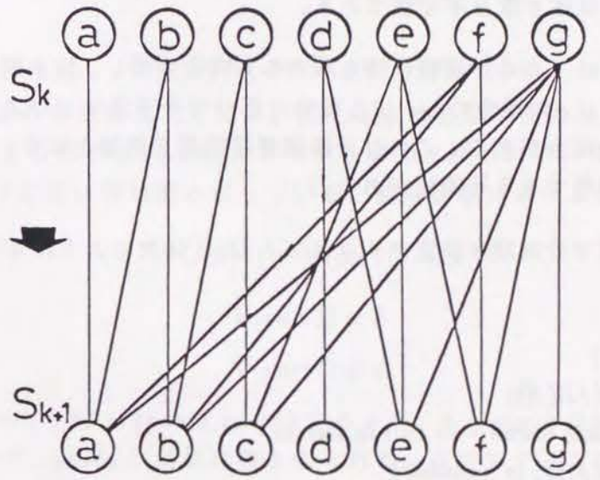


図 7.4: S_k から S_{k+1} への遷移

5. 1/2 平面：

$$S_k = \{\mathbf{v} | (\mathbf{v} \cdot \mathbf{p} = 0) \wedge (\mathbf{v} \cdot \mathbf{p}_s \geq 0)\}$$

6. 平面片：

$$S_k = \{\mathbf{v} | (\mathbf{v} \cdot \mathbf{p} = 0) \wedge (\mathbf{v} \cdot \mathbf{p}_{s_1} \geq 0) \wedge (\mathbf{v} \cdot \mathbf{p}_{s_2} \geq 0)\}$$

7. 半空間の積集合：

$$S_k = \bigcap_j \mathbf{V}_j$$

S_kからS_{k+1}への遷移は図7.4のように表される。最終的に、部品の移動可能方向$\mathbf{V}$はS_nとして得られる。

7.4 分解順序決定アルゴリズム

組立順序の決定は、分解順序の逆順として得られる。分解順序は、抜き取り可能な部品を順番に抜き取ることにより生成される。部品の抜き取り可能性は次のように判定する。

1. 前節で述べた抜き取り方向の決定法により、抜き取り方向$\mathbf{V}$を求める。
2. $\mathbf{V} = \phi$の場合は抜き取り不可能である。
3. $\mathbf{V} \neq \phi$の場合は$\mathbf{V}$の中で実際に抜き取れる方向を探索し、抜き取り可能かどうか調べる。($\mathbf{V} \neq \phi$の場合でも、少し離れた部分での干渉などによって、実際には抜き取れない場合がある。これは干渉回避経路発見問題となり、文献[8,9,10]などの結果を利用できる可能性がある。)

以上の準備の基で分解順序決定アルゴリズムDSPは次のようになる。

Algorithm DSP

1) $IR(A) \leftarrow f_{IR}(A)$
2) $MD(A) \leftarrow f_{MD}(IR(A))$
   if no movable parts found $\longrightarrow$ end in failure
   otherwise $M = \{P_i | P_i \text{ is movable}\}$
3) select the part $P_i \in M$
4) reconstruct the sub-assembly model without the part P_i
   $T \leftarrow f_{IRE}(A - P_i)$

   if $f_{SC}(A - P_i, T)$ is failure ($A - P_i$ is unstable)
   $\longrightarrow$ select another part $P_i \in M$
      if no another part P_i $\longrightarrow$ end in failure
      otherwise go to 4)
   otherwise $A \leftarrow A - P_i$
      if $A = \phi$ $\longrightarrow$ end in success
      otherwise $IR(A) \leftarrow T$
         go to 2)

ただし、アルゴリズムDSPにおいて、Aはn個の部品よりなる組立製品モデル、$f_{IR}(A)$はAのすべての2部品間の干渉認識を実行する関数、$f_{MD}(IR(A))$はAのすべての部品の移動可能方向の集合を求め、上で述べたことも考慮して抜き取り可能方向を求める関数、$f_{IRE}(A - P_i)$はAから部品P_iを取り去った後の干渉認識情報を求める関数(P_iに関する接触領域を消去することにより得られる)、$f_{SC}(A - P_i, T)$は組立構造物の安定性を判定する関数である。

7.5 実験例

図7.5に移動可能方向および分解順序の決定例を示す。図7.5(a)は4部品よりなる組立製品モデル

$$A = \{part1, part2, part3, part4\}$$

とTIPS-1[2]による形状定義文である。図7.5(b)は移動可能方向および分解順序の決定結果である。この例では、$part3$がテーブルの上に置かれていることを仮定しているので、$part3$の下底面は接触面となっている。初期状態における移動可能方向は、

$$\mathbf{V}(part1) = \mathbf{V}(part2) = \phi$$

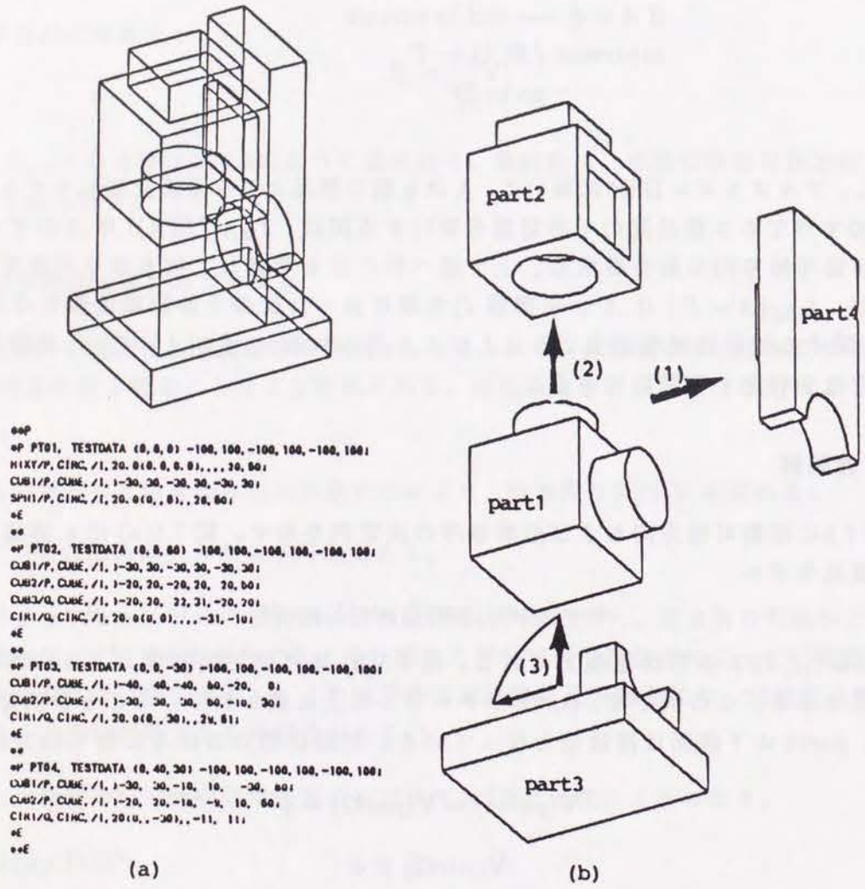
$$\mathbf{V}(part3) \neq \phi$$

$$\mathbf{V}(part4) \neq \phi$$

となる。したがって、$M = \{part3, part4\}$であるが、もし$part3$を抜き取れば$A - part3$は安定でないので、$part4$が最初に抜き取られる。次のステップでは、

$$A = \{part1, part2, part4\}$$

$$M = \{part2, part3\}$$




```
#P
#P PT01, TESTDATA (0, 0, 0) -100, 100, -100, 100, -100, 100;
HIXY/P, CIRC, /1, 20. 0(0. 0, 0. 0), , , 20, 50;
CUB1/P, CUBE, /1, )-30, 30, -30, 30, -30, 30;
SPH1/P, CIRC, /1, 20. 0(0, , 0), , 20, 50;
#E
#
#P PT02, TESTDATA (0, 0, 50) -100, 100, -100, 100, -100, 100;
CUB1/P, CUBE, /1, )-30, 30, -30, 30, -30, 30;
CUB2/P, CUBE, /1, )-20, 20, -20, 20, 20, 50;
CUB3/Q, CUBE, /1, )-20, 20, 22, 31, -20, 20;
CIR1/Q, CIRC, /1, 20. 0(0, 0), , , , -31, -10;
#E
#
#P PT03, TESTDATA (0, 0, -30) -100, 100, -100, 100, -100, 100;
CUB1/P, CUBE, /1, )-40, 40, -40, 50, -20, 0;
CUB2/P, CUBE, /1, )-30, 30, 30, 50, -10, 30;
CIR1/Q, CIRC, /1, 20. 0(0, , 30), , 24, 61;
#E
#
#P PT04, TESTDATA (0, 40, 30) -100, 100, -100, 100, -100, 100;
CUB1/P, CUBE, /1, )-30, 30, -10, 10, -30, 50;
CUB2/P, CUBE, /1, )-20, 20, -10, -9, 10, 50;
CIR1/Q, CIRC, /1, 20(0, , -30), , -11, 11;
#E
#
#E
```

図 7.5: 移動可能方向と分解順序の生成例

となり、同様の過程を経て *part2* が抜き取られる。さらに、3 番目に *part1* が抜き取られ、最後に *part3* が抜き取られて $A = \phi$ となり分解順序の生成が終了する。

7.6 まとめ

本章では、組立製品モデルに対して各部品間の干渉認識を行うことにより、組立順序自動決定法として、

1. 各部品の抜き取り可能方向の導出法
2. 分解順序決定アルゴリズム DSP

を示した。アルゴリズム DSP 中で述べた組立構造物の安定性の判定法は現段階では確立されていないが、干渉認識法とソリッドモデルによるマスプロパティの計算などによって可能になると思われる。

参考文献

[1] 渡部広一、嘉数侑昇、沖野教郎: ソリッド形状モデリングにおける CSG から B-Reps への解析的変換の研究、精密工学会誌、**53** 、2(1987)315.

[2] 沖野教郎、嘉数侑昇、久保　洋 : 自動設計プロセサ TIPS-1 の開発、精密機械、**44**, 3 (1978)371.

[3] 渡部広一、嘉数侑昇、沖野教郎 : 3 次元形状モデル間の干渉認識に関する研究、精密工学会誌, **53**, 6 (1987)965.

[4] 北島克寛、吉川弘之 : 階層的ネットワークモデルに基づく対話型機械設計システム HIMADES-1 の開発、精密機械、**47**, 12 (1981)50.

[5] 関口博、小島俊雄、井上久仁子、本多庸悟 : 回転機能部品の部品展開手法に関する研究—組立構造の表現法と組立・分解手順の生成への応用、　精密工学会誌、**51**, 2 (1985)359.

[6] 関口博、今村聡、小島俊雄、井上久仁子 : 回転機能部品の部品展開手法に関する研究—（第 2 報）組立・分解手順の生成とその規則化、　精密工学会誌、**53**, 8 (1985)1183.

[7] L.Laperriere and H.A.EIMaraghy : "Automatic Generation of a Robotic Assembly Sequence", Proc. of the 1st ASME Conference on Flexible Assembly Systems, DE-Vol.20, (1989)15.

[8] T. Lozano-Perez and M.A.Wesley : "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles", Communications of the ACM, Vol.22, No.10, (1979)560.

[9] J.F.Canny : "The Complexity of Robot Motion Planning", The MIT Press, MIT, (1987).

[10] 木村秋広、渡部広一、沖野教郎：干渉回避問題に対する解析的アプローチ、1989年度精密工学会秋季大会学術講演会論文集、(1989)559.

[11] H.Watabe, N.Miyoshi, Y.Kakazu and N.Okino : "Interference Recognition among 3D Solid Models for Assembly Planning", Proc. of the 1st ASME Conference on Flexible Assembly Systems, DE-Vol.20, (1989)79.

第 8 章

CAD/CAM システム構築環境

8.1 はじめに

本章は、形状モデルをベースとしてCAD/CAM システムを構築するときの環境そのものに対して新しい提案をする。前章までの各章において、基本形状曲面の相貫曲線解析解、フィレットのCSG 表現法、CSG からB-Reps への解析的変換、金型 CAD用反転形状の自動生成、ソリッド形状モデル間の干渉認識、組立順序の自動生成について論じてきた。これらの各章で開発された各手法などをプログラム化し、さらに、目的に応じて各種のプログラムを作成し、それらを結合することによってCAD/CAM システムを構築するわけであるが、その作業は大変なものであり、また、いったんできたものを修正・管理することも非常に困難が付きまとう。

また現在稼働中の多くのCAD/CAM システムを見ても、ユーザにとって使い良く満足のいくシステムは少ないようである。CAD/CAM システムと一口に言っても、それは膨大な内容を含んでおり、全てを網羅して誰にとっても有効なシステムを構築することはほとんど不可能と言っても良いであろう。従って、実際に有用なシステムを手にいれたければ、そのシステムを実際に使う人が、目的にあったCAD/CAM システムを自前で作成するのが一番ということになる。しかし、そのようなユーザが自分でシステムを一から作ろうとすると莫大な労力と時間を必要とする。

そこで考えられる方法として、既に存在するシステムを修正して、より目的にあったものにカスタマイズするか、CAD/CAM システム構築用部品を組み合わせて自分用（自社用）システムを構成する方法などがある。前者の方法は一般に非常に困難である。なぜなら、既に存在するシステムは、大抵の場合、全体で一つのものとなっており、ある部分を変更すると他の部分に影響が及ぶといったことが起こりやすく、また、どこをどう修正すれば良いのかを発見することが非常に困難である。

後者の方法は、ユーザが必要な部品を選択して結合させることにより、そのユーザの目的にあったシステムを構成する方法である。もし、与えられた部品がそのまま使えず、部品の修正をする場合でも、小さな部分に限って考えることが出来るので、かなり容易になると思われる。

このとき、各部品同志が結合するだけで全体を制御する部分を新たに付け加えなくてもシステムが出来上がることが望ましい。全体を制御する部分がないと言うことは、実行に際して、各プログラム部品は上位の部分からコール（実行を要請）されることはないということになる。したがって、各プログラム部品が自分の方から積極的に実行を開始しなければならない。このようなプログラム部品のことを自律駆動型プログラムモジュール（ADPM）と呼ぶことにする。

本章は、自分用のCAD/CAM システムを構築しようとしているユーザに、CAD/CAM 用部品を作成・テストしながら目的のシステムを構成するための環境を提供しようとするものである。前章までの各プログラムを自律駆動型プログラムモジュールとすることによって、非常に自由度の高い CAD/CAM システムを構成できる可能性を示す。

ここで提案するADPM は、概念的には、インテリジェントCAD/CAM 用モデリングエレメントとして提案されているモデロン [1,2] に含まれると考えることができる。すなわち、モデロンの一部の機能を実現したものとしてとらえることも可能である。

以下の各節において、ADPM の動き、ADPM の構造、ADPM のオペレーション、P-ADPM（既存プログラムモジュールの組み込み）、ADPM を実行する場としてのインタープリタの開発、ADPM による CAD/CAM システムの構成例について順に説明する。

8.2 ADPM の動き（外からみた様子）

自律駆動型プログラムモジュール（ADPM）とは、プログラムモジュールが自分から積極的に実行状態にはいる、すなわち、発火するようなプログラムモジュールのことを指す。しかし、各プログラムモジュールが勝手気ままに実行状態に入っていては、目的の答えが得られるはずはないし、収拾のつかない状態になるであろう。そこで、各プログラムモジュールに外からみると図8.1に示すような要求駆動的な動きをさせるものとする。すなわち、

1. 自分が出来る仕事（出力引数）への要求があるかどうか見る。
2. その要求があれば、要求されている出力引数を計算するための入力引数を探す。もしあれば、結果を計算して環境（ワーキングスペース）に返す。なければ、入力引数を計算するように要求を出す。

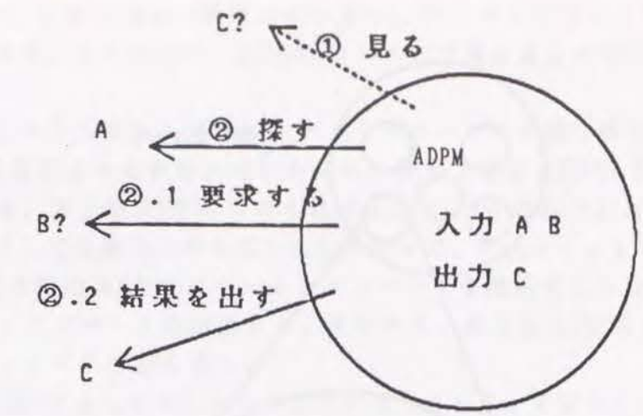


図 8.1: ADPM の動き

図8.1の例で見ると、このADPM は自分は入力引数としてA とB があれば、出力引数C を計算できることを知っている。そこで、ワーキングスペースにC？という要求があると、それに反応して、C を計算しようとする。そのためにA，B を取り込もうとするが、ワーキングスペースにはA しかないので、B？という要求をワーキングスペースに出し、他の ADPM がB を出力してくれるのを待つことになる。

8.3 ADPM の階層構造

前節で述べたような動きをするADPM を構成するために、ADPM システム全体としての構造を決定する。そのために以下の点を考慮する。

1. ADPM にはその動作環境としてのワーキングスペースが必要である。
2. ADPM にはプログラムモジュールとしてのプログラム記述部分が必要となる。
3. 多数の ADPM が同じ動作環境内に存在すると効率の面などで不都合が発生することが考えられるので、グループ毎にまとめられた方が良い。
4. 既に存在するプログラムモジュールは有効に活用したい。

そこで、ADPM の構造として図8.2に示すものを考える。これは、標準ADPM については文献 [1] で提案されているモデロンの構造と同様である。すなわち、標準 ADPM はオペレーション、オブジェクト、ワーキングスペースよりなる。オペレーションはプログラムモジュールとしてのプログラム部分で、個数に制限はない。オブジェクトは下

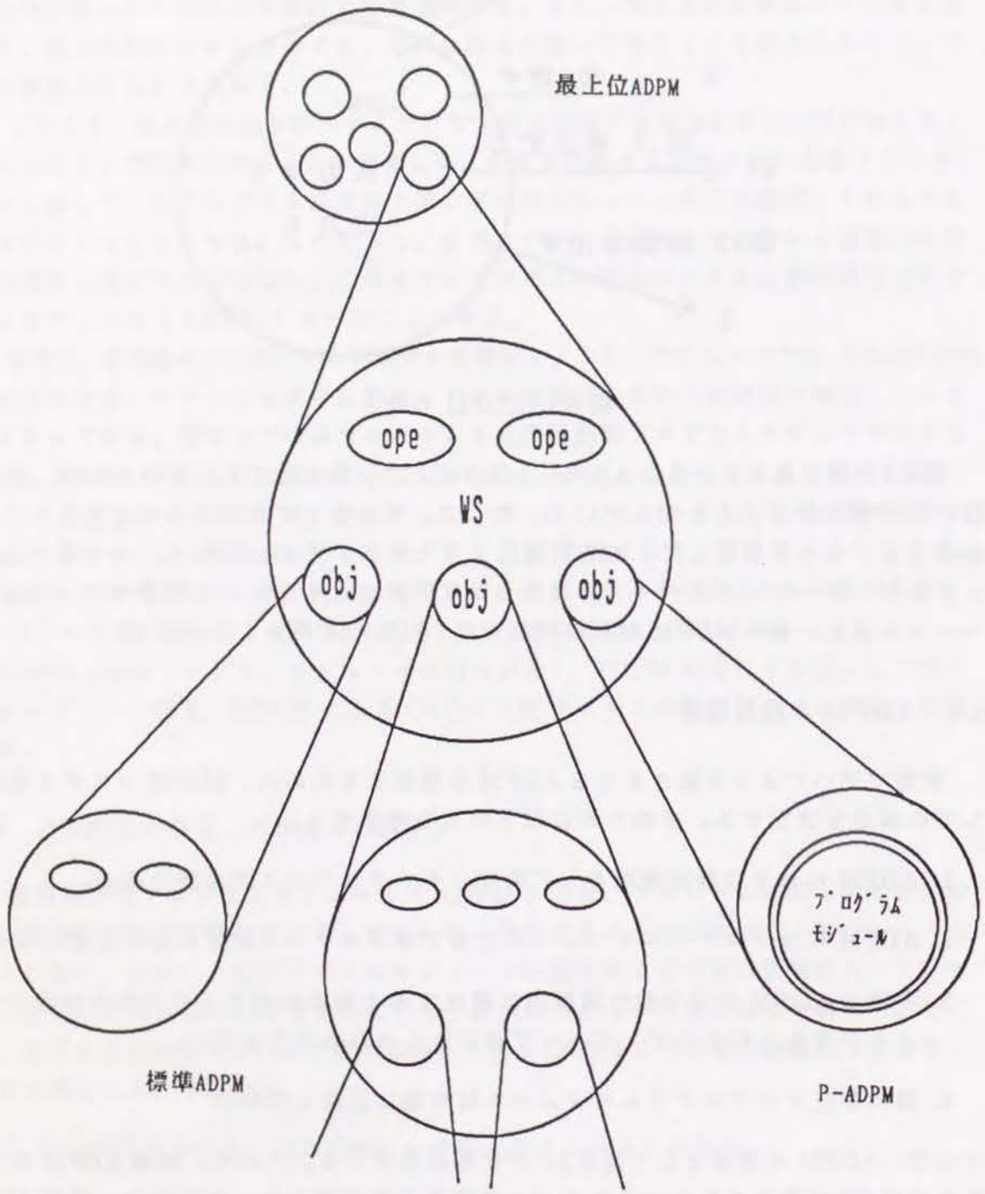


図 8.2: ADPM の階層構造

位の ADPM で、これも個数に制限はおかない。ワーキングスペースは下位の ADPM の動作環境となる。したがって、ADPM はグループ毎にまとめることによって階層構造を成す。

各 ADPM はその上位の ADPM のワーキングスペースを動作環境とするが、無限に上位 ADPM を存在させるわけにはいかない。最も上位の ADPM を最上位 ADPM と呼ぶことにする。最上位 ADPM にはもはや上位 ADPM は存在しないので、プログラムモジュールとしての動作は行わない。したがって、オペレーションは不要となる。最上位 ADPM は下位の ADPM にワーキングスペースを提供すると同時に、ユーザに対してもワーキングスペースを提供する。すなわち、最上位 ADPM はユーザと ADPM のインターフェイスの役割を持つ。

再下位の ADPM としては二通りのものが考えられる。オブジェクト、すなわち、下位の ADPM を持たないオペレーションとワーキングスペースのみよりなる ADPM が一つ。もう一つは、P-ADPM と呼ばれるもので、プログラムモジュールとしてのオブジェクトのみよりなる ADPM である。P-ADPM のオブジェクトは既存のプログラムモジュールなので、それ以下の ADPM は存在しないことになる。

8.4 ADPM のオペレーション

前節で示したように、各 ADPM はオペレーション、オブジェクト、ワーキングスペースより構成されるが、オブジェクトは P-ADPM 以外は下位の ADPM であり、ワーキングスペースは単なるデータ領域であるので、ADPM の構造の中でオペレーションが中心的な役割を果たす。そこで本節において、ADPM の自律駆動性を達成するためのオペレーションの機能について述べる。

8.4.1 オペレーションの機能分解

ADPM に 8.2節で述べたような動きをさせるためにオペレーションを図 8.3のように構成する。すなわち、オペレーションを 3 つの部分（入力部、本体、出力部）に分け、機能分解をする。入力部の仕事は、仕事の受注、入力データの要求、入力データの取り込みである。本体は、下位の ADPM に仕事を発注する。ただし、下位の ADPM も自律駆動的なので、どの ADPM に何をさせるかを指定する必要はなく、何をして欲しいかの注文をするだけでよい。出力部は、本体によって出された発注に対する答えを収集し、出力引数を計算して上位の環境に返す。

8.4.2 オペレーションの状態遷移

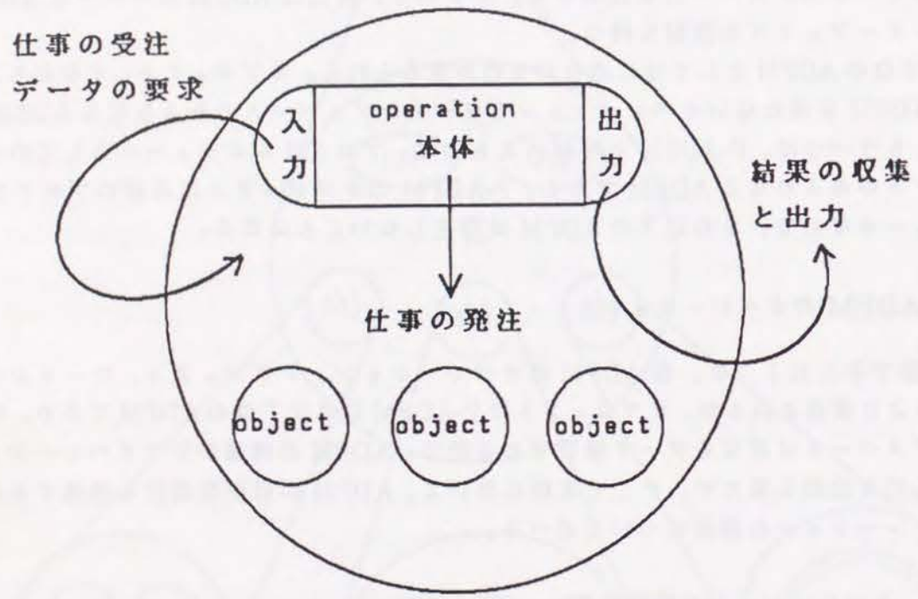


図 8.3: オペレーションの構造

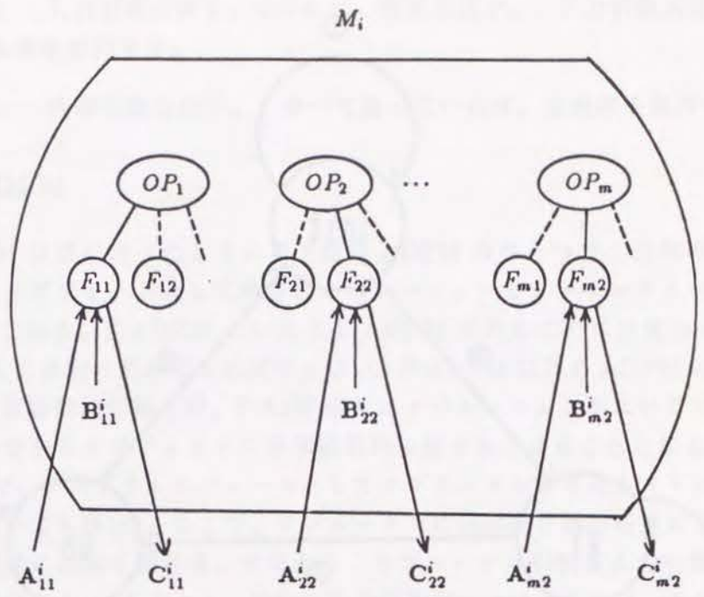


図 8.4: オペレーションの複数機能

一つのオペレーションには複数の機能（function）を持たせる。言い替えれば、一つのオペレーションには、複数個の入出力パターンを持たせ、状況に応じて適当な機能を発現させるようにする（図 8.4）。従って、状況に応じて、どの機能を発現させるかを選択しなければならない。選択において評価すべき点は、

1. 要求に最もよく合う、すなわち、より多くの要求に応えられる機能を選ぶ。

2. 同程度に要求に答えられる機能が複数個ある場合、その機能が必要とするものがより多く揃っているものを選ぶ。すなわち、入力引数がより多く揃っている機能を選択する。

である。

一度機能が選択されたならば、その機能が答えを出すまでは、そのオペレーションの機能は固定される。（動的に機能を選択し直すことも考えられるが、現段階では考えない。）

このようにオペレーションの機能がまだ決まっていない状態を初期状態（init）と呼ぶ。機能が固定されて初期状態を脱すると、オペレーションは入力待状態（wi）ま

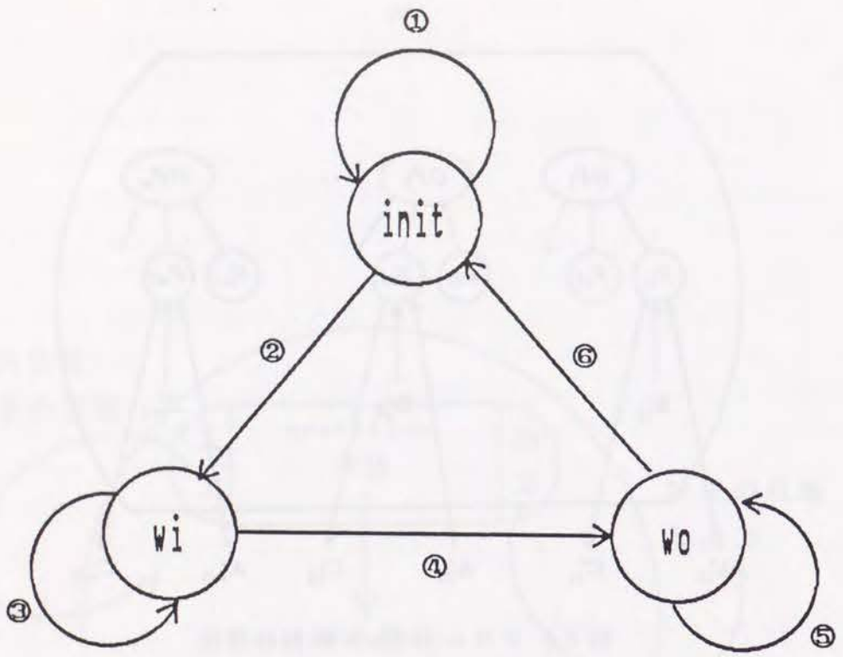


図 8.5: オペレーションの状態遷移

たは、出力待状態（ wo ）に入る。そして、出力待状態から出力引数を計算し終わると、オペレーションはまた初期状態に戻る。

ここでオペレーションの状態遷移をまとめると次のようになる。（図 8.5）

1. init ⟶ init :要求にマッチする機能がない場合
2. init ⟶ wi :要求にマッチする機能があった場合（機能固定）
3. wi ⟶ wi :入力引数が揃わない場合
4. wi ⟶ wo :入力引数が揃った場合（本体の実行）
5. wo ⟶ wo :内部引数が揃わない場合
6. wo ⟶ init :内部引数が揃った場合（出力部の実行、出力）

また、各状態においてオペレーションのなすべき仕事は、以下のようになる。

初期状態: 自分がなすべき仕事（要求）を探し、その要求に最もよくマッチする機能（ function ）を決定する。

入力待状態: 入力引数を探す。なければ、要求を出す。 入力引数が全部揃っていれば、本体を実行する。

出力待状態: 内部引数を探す。 すべて揃っていれば、出力部を実行する。

8.5 P-ADPM

P-ADPM は既に述べたように再下位の ADPM のひとつで、既存のプログラムモジュールをオブジェクトとして持ち、オペレーションとワーキングスペースを持たない ADPM である。P-ADPM といえども ADPM であることには変わりはないので、P-ADPM も自律的に動かなければならない。P-ADPM 以外の ADPM はそのオペレーションが自律駆動的に動くが、P-ADPM にはオペレーションがないので、プログラムモジュールであるオブジェクトに自律駆動的な動きをさせることとなる。

ここでは、プログラムモジュールとしてサブルーチンを考える（下位のサブルーチンを呼んでいても良い）。そこで、サブルーチンに自律的な動きをさせるために、サブルーチン記述に制限を設ける。すなわち、サブルーチン引数を入力引数か出力引数に分ける必要がある。（このとき、純粋な出力引数が一つでもあれば、その他は入出力引数でもよい。）そうすることによって、サブルーチンにオペレーションと同様な動きをさせることができる。ただし違いは、

- 機能が一つしかないこと
- 下位のオブジェクトが存在しないことにより、8.4.2節における 5,6 が不要になり 4 において入力引数が揃った場合、サブルーチン本体を実行することになること

である。

8.6 ADPM インタープリタ

8.2節から 8.5節で述べた考えに従って、ADPM インタープリタを Lisp 言語を使用して作成した。8.4、8.5節では、主に個々の ADPM の動き、およびその制御法について述べたので、本節では Lisp 上における ADPM の内部構造、および、ADPM インタープリタによる全体の制御について述べる。

8.6.1 ADPM 構造体

インタープリタで実行される ADPM の内部構造は以下のような Lisp の構造体である。

```
(defstruct ADPM
    PrimitiveFlag
    ParentADPM
    ObjectsList
    OperationsDefine
    ObjectDefine
    WorkingSpace)
```

ここで、PrimitiveFlag は P-ADPM かどうかの判定用フラグ。ParentADPM は上位 ADPM 名。ObjectsList は下位 ADPM 名を要素とするリスト。WorkingSpace はリストである。以下、OperationsDefine 、ObjectDefine について説明する。

OperationsDefine の詳細

OperationsDefine は OperationDefine を要素とするリストである。すなわち、

```
OperationsDefine
= (OperationDefine1 OperationDefine2 ... OperationDefineN)
```

各 OperationDefine は以下のようなリストである。

```
( [オペレーション名]
  (status [状態])
  (FunctionsDefine    [FunctionsDefine])
  (FiredFunction [現在発火している FunctionDefine]))
```

ここで、[FunctionsDefine] は FunctionDefine を要素とするリストである。すなわち、

```
[FunctionsDefine]
=    (FunctionDefine1 FunctionDefine2 ... FunctionDefineN)
```

FunctionDefine を複数個定義できるということは、複数通りの入出力パターンを用意できるということに対応している。

各 FunctionDefine は以下のようなリストである。

```
( (BodyFunction        [本体関数名])
  (OutputFunction      [出力関数名])
  (InputArgument       [入力引数名のリスト])
  (InsideArgument      [内部引数のリスト])
  (OutputArgument      [出力引数名のリスト]))
```


ここで、[本体関数名] 、[出力関数名] はユーザ定義の Lisp 関数の名前である。また各引数定義は、次のような書式で行う。

[入力引数名のリスト] = (form1 form2 ... formN)　　$N \geq 0$
[内部引数名のリスト] = (form1 form2 ... formM)　　$M \geq 0$
[出力引数名のリスト] = (arg1 arg2 ... argL)　　$L \geq 1$

ただし、formI は以下の 3 パターンのいずれかの書き方が可能である。

1. (argI [要求先 ADPM 名]　nil　...)
2. (argI [要求先 ADPM 名])
3. argI

1 の... の部分には何を幾つ書いてもよい。argI は任意の変数名。[要求先 ADPM 名] はその引数の値を計算して欲しい ADPM の名前で、以下のものが使える。

```
@mother      （上位 ADPM ）
@!mother     （上位 ADPM 以外）
@brother     （同じ階層に属する ADPM の一つ）
@brothers    （同じ階層に属する全ての ADPM ）
@self        （自分）
@!self       （自分以外の ADPM ）
@child       （下位 ADPM の一つ）
@children    （全ての下位 ADPM ）
nil          （任意の ADPM ）
実 ADPM 名
```

[要求先 ADPM 名] を指定することに関しては、これをすべて実 ADPM 名で指定するとオブジェクト指向におけるメッセージパッシングのようになり、名指しで実行を促されることになるので、自律駆動型プログラムモジュールとは相反するものになる可能性がある。ADPM 作成時は、なるべく実 ADPM 名を指定しないことが基本である。

ObjectDefine の詳細

ObjectDefine は以下のようなリストである。

```
( [オブジェクト名] [ファイル名]
  ( ([引数名] [入出力パタン] [引数の型])
    ([      ] [           ] [      ])
                    :
                    :
    ([      ] [           ] [      ])))
```

[オブジェクト名] は Fortran サブルーチンの名前。[ファイル名] はサブルーチンの実行モジュールが入っているファイルの名前である。ここで、サブルーチンが更にサブルーチンを呼んでいるような場合でも [ファイル名] で指定されているファイルにそれらの呼ばれているサブルーチンの実行モジュールが入っていれば問題なく実行できる [3]。

引数定義は以下のようになる。

引数名: 引数名には **OperationsDefine の詳細**で述べた form が使える。

引数の型: 引数の型は、以下のものが使用できる。

integer	4 バイト整数型
real	4 バイト実数型
(array-integer [次元・サイズ])	4 バイト整数配列
(array-real [次元・サイズ])	4 バイト実数配列

入出力パタン: 入出力パタンは、以下のものが使用可能である。

in	入力引数
out	出力引数
inout	入出力引数

8.6.2 全体の制御

各 ADPM は Lisp の構造体として定義されているので、いわば単なるデータの一種である。そこで、ADPM にユーザから見て自律駆動的な動きをさせるために、ADPM インタープリタは全体の制御をする必要がある。そのために、実行候補 ADPM の登録を行い、登録された ADPM を順番に実行するという方式を提案する。

実行候補 ADPM の登録

実行候補 ADPM は、各 ADPM がワーキングスペースに要求やデータを出したときに、以下のルールにしたがってシステム変数*modelon-order*に登録される。*modelon-order*はリストである。

1. ワーキングスペースに要求を出したときは、要求を出された ADPM（当 ADPM）の名前を*modelon-order*に左から追加し、さらに、当 ADPM の全ての下位 ADPM 名を*modelon-order*に左から追加する。
2. ワーキングスペースにデータを出したときは、データを出された ADPM の下位 ADPM 名全てを*modelon-order*に左から追加する。
3. ワーキングスペースに要求を出そうとしたとき、既にデータがあれば要求は出さないが（要求を出すとデータが消えてしまうため）出したと同じ扱いにして、ルール 1 に従う。
4. 追加しようとする ADPM 名が既に*modelon-order*内にあれば、その ADPM 名は追加しない。

ただし、1、3 において同じ要求が既にあれば（前回自分が出した要求がそのまま残っている場合など）改めて要求を出すことはしない。従って実行候補 ADPM の登録も行われない。

実行アルゴリズム

ADPM インタープリタは以下のようなアルゴリズムで ADPM の実行を制御する。

0. ユーザが最上位 ADPM（オペレーションがなく下位の ADPM とユーザにそのワーキングスペースだけを提供する ADPM である。）に要求及びデータを設定する。

1. *modelon-order*が nil ならば、終了。
そうでなければ、対象 ADPM をトップ ADPM（*modelon-order*の第一要素）に設定する。
また、*modelon-order*からトップ ADPM を抜いておく。

2. 対象 ADPM が P-ADPM でなければ、対象 ADPM のオペレーションを順次実行する。
対象 ADPM が P-ADPM ならば、P-ADPM を実行する。

3. 1 へ行く。

8.6.3 実行の停止性

以上のアルゴリズムによって制御されるADPMの実行は、必ず停止することを示すことができる。すなわち、*modelon-order*は最終的にはnilになる。このことは、次の1,2により明らかである。

1. 各ADPM（オペレーション）の出せる要求は有限個であり、同じ要求は2度と出さない。これは、(1) 一度要求に答えてデータを出力した場合はそのデータは残ったままになっているので、要求は出さないことと、(2) 一度出した要求が要求のまま残っているときは何もしないこと による。

2. 各ADPM（オペレーション）は要求がなければ発火しない。

しかし、始めにユーザが設定した要求に答えられるかどうかは保証されない。これは、各ADPMがどのように作られているかと、どのように結合されているかに依存する。ただ、目的の答えが得られなかった場合には、不足したデータに対する要求がワーキングスペースに残っているので、ユーザがその情報を依りどころにしてADPMを修正していくことができる。

8.7 自律駆動型プログラムモジュールによるCAD/CAMシステムの構成

本節では、ADPMインタープリタをCAD/CAMシステム構築環境としてとらえ、CAD/CAMシステム用部品としてADPMを適用した例を示す。

8.7.1 部品形状設計用システム

本節では、一つの部品形状設計用システムについて述べる。図8.6にその構成例を示す。このシステムは、機能的にはCAD/CAMシステムTIPS-1[4]に準拠している。一番外側の円で表されるPART ADPMがいま設計しようとしている部品モデルであり、同時にそれが部品形状設計用システムとなる。PART ADPMは、いくつかの下位ADPMとワーキングスペースよりなる。各下位ADPMには、例えば以下のような機能を持たせることになる。

shape_of_part　いくつかのパラメータを受け取って形状定義を行うADPM。

t1pr　形状定義を受け取ってCSGデータを作成するADPM。

eagle　CSGデータを受け取ってB-Repsデータを作るADPM。

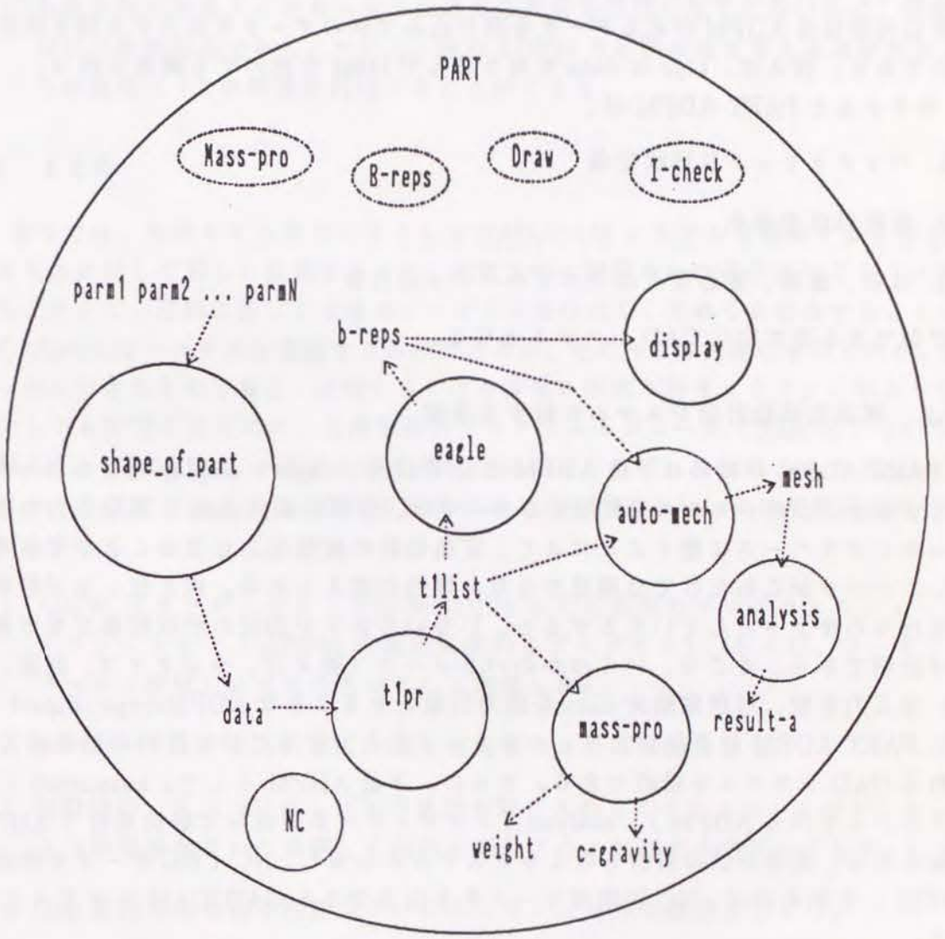


図 8.6: 部品形状設計用システムの構成

mass_property　CSG データを受け取ってマス・プロパティ計算を行う ADPM 。

display　CSG 、B-Reps データを受け取って形状の画面表示を行う ADPM 。

また、parm1 、data 、weight 等は PART ADPM のワーキングスペース内に存在する変数で、それらは実行の過程によって要求になったりデータになったりする。図中、破線の矢印は各 ADPM があるデータを取り込んで別のデータを出力する様子を示したものである。例えば、t1pr は data を取り込んで t1list を出力する機能を持つ。

そうすると PART ADPM は、

1. パラメトリックな形状定義
2. 定義形状の表示
3. 体積、重量、重心などのマスプロパティの計算

などができる基本的な CAD システムとなる。

8.7.2 部品形状設計用システムに対する考察

PART ADPM が初めは下位 ADPM として t1pr 、eagle 、display だけを持っていたとすると、このシステムを利用するユーザは、形状定義文 data と図形出力の要求をワーキングスペースに置くことにより、定義形状の画像出力を見ることができる。しかし、ユーザがこれだけでは満足できない場合が考えられる。例えば、ネジ形状の表示を行なわせようとしていたとすると、いちいち各ネジ形状の形状定義文を定義するのは面倒である。そこで、いくつかのパラメータ（例えば、ネジタイプ、直径、長さ等）を入力引数、形状定義文 data を出力引数とするような ADPMshape_of_part を作成し PART ADPM に追加すれば、パラメータを入力するだけで目的の画像出力が得られる CAD システムを構成できる。さらに、下位 ADPM として、auto_mesh（解析用メッシュを作る ADPM ）、analysis（メッシュデータを使って解析を行う ADPM ）を加えれば、定義形状の解析も行えるシステムとなるし、NC（ NC データを生成する ADPM ）を加えれば、NC 切削用データをも出力できる CAD/CAM システムとなりうる。

ここで、ADPM を CAD/CAM システム構築用部品としてシステムを構築する際の特徴をあげると以下のようになる。

1. ユーザが入力すべきデータをはっきり覚えていなくてもシステムが教えてくれる。例えば、上の例でユーザが入力すべきパラメータを与えずに実行したとすると、システムからそれらのパラメータを要求される。

2. 各部品が ADPM としての体裁さえ整っていれば、いつでも実行可能である。従来のサブルーチンからシステムを構成する場合は、各サブルーチンをテストするためにテスト用のメインルーチンを作成しなければならなかったが、そのようなことは必要ない。
3. 全体を制御するメインルーチンに当たる部分を作成しなくて良いので、機能の追加が比較的容易である。ただし、他の ADPM との整合性を考える必要があるが、その過程で 1,2 の特徴を利用することができる。

8.8 まとめ

本章では、形状モデルをベースとして CAD/CAM システムを構築するときの環境そのものに対して新しい提案を行った。前章までに開発された各手法などをプログラム化し、さらに、目的に応じて各種のプログラムを作成し、それらを結合することによって CAD/CAM システムを構築するわけであるが、その作業は大変なものであり、また、いったんできたものを修正・管理することも非常に困難が付きまとう。このような現状を少しでも打開するために、自律駆動型プログラムモジュール（ ADPM ）という概念を導入し、CAD/CAM システム構築用部品の集合として、目的にあった CAD/CAM システムを構成できるような環境を提供することを目標とし、その基礎研究を行った。すなわち、

1. ADPM はオブジェクト（下位の ADPM または既存のプログラムモジュール）、オペレーション（ ADPM の働きを決めるプログラム）、および、ワーキングスペース（環境）よりなるものとして定義した。
2. オペレーションを入力部、本体、出力部の 3 つに機能分解した。
3. 初期状態、入力待状態と出力待状態を取り入れることにより、オペレーションの 3 つの部分が互いに連携して処理が進むように制御する方式を示した。
4. Lisp 言語による自律駆動型プログラムモジュールの構造を示した。
5. 自律駆動型プログラムモジュールを実行させるインタープリタの制御方式を示した。

CAD/CAM 用の各プログラムを自律駆動型プログラムモジュールとすることによって、非常に自由度の高い CAD/CAM システムを構成できる可能性を示した。

参考文献

[1] 沖野教郎：インテリジェント CAD/CAM 用モデリングエレメント「モデロン」とその支援システムの開発, 1989 年度精密工学会春期大会論文集, (1989)19.

[2] N. Okino : Toward Synthetic CAD—A New Paradigm, Proc. of the USA-Japan Symposium on Flexible Automation, Vol.1, (1988) 11.

[3] apollo Domain/CommonLISP User's Guide, apollo Computer Inc. (1986).

[4] 沖野教郎, 嘉数侑昇, 久保　洋：自動設計プロセッサ TIPS-1 の開発, 精密機械, 44, 3 (1978)371.

[5] H.Watabe, Y.Ide and N.Okino : "Robot Task Planning by Autonomous Drive Program Modules—Automatic Generation of Grasping Locations", Proc. of the 2nd ASME Conference on Flexible Assembly Systems, DE-Vol.28, (1990)53.

第 9 章

結論

本論文は、3次元ソリッドモデルをベースとする高精度なCAD/CAMシステムを構築するために必要不可欠となる高精度で高速な形状モデリングの手法についての研究をまとめたものである。すなわち、設計・製造のための形状情報をソリッドモデルとして計算機内に格納するためのモデリング手法の開発と、その格納されたソリッドモデルから、目的の製品に関する情報、および、製品の設計・製造過程のシミュレーションに必要となる情報を抽出するための形状モデリング手法の開発を行った。具体的には、ソリッドモデルとして代表的なCSGとB-Repsを採用し、CSGからB-Repsへの変換手法を開発することによって、互いの長所を生かし短所を補う形のCSG/B-Reps二重構造モデルの構築を行った。またその変換手法には、従来とられていたCSGプリミティブの多面体化による近似手法ではなく、CSGプリミティブを構成する曲面をそのまま扱う解析的な方法をとることによって、高精度で高機能な形状モデルとなった。さらに、そのようなCSG/B-Reps二重構造モデルを利用したいくつかのシミュレーションプログラムを開発することによって、高精度CAD/CAM構築のための形状モデリングの有効性を示した。

以下、幾つかの項目毎にまとめる。

1. 高精度な形状モデリングのための中心的な道具を用意するために、曲面間の相貫曲線の解析的厳密解を求めた。一般の曲面間の相貫曲線を解析的に求めることは困難であるが、CSGプリミティブを構成する曲面は、ほとんどが平面、2次曲面、トーラスであることを考慮して、それらの曲面間の相貫曲線を解析的に求めた。すなわち、

   (a) 曲面間の相貫曲線解析解の一般的な求め方として、陰関数法、パラメータ法およびパラメータ／陰関数法を論じた。

(b) 平面と2次曲面との相貫曲線解析解を陰関数法で求めた。

(c) 2次曲面同志の相貫曲線解析解をパラメータ法で求めた。

(d) トーラスと球との相貫曲線解析解をパラメータ法で求めた。

(e) トーラスと線織面との相貫曲線解析解をパラメータ／陰関数法で求めた。

(f) トーラス同志の相貫曲線解析解をパラメータ法で求めた。

(第2章)

2. CSG表現におけるソリッドモデルの高精度化のために、CSGプリミティブとしてのフィレット曲面の創成法を示した。すなわち、

(a) フィレット曲面創成問題をスウィープ曲面創成問題に置換して扱う方法を示した。この場合、3次元空間の問題をスウィープ断面の2次元問題に帰着させて扱う点に特徴がある。

(b) スウィープソリッド用ペナルティ関数の設定方法を示した。

(c) いくつかの具体例によって計算機実験を行い、フィレット曲面のソリッドモデルが創成できることを明らかにした。そして、創成されるフィレットボリュームはCSG表現法におけるプリミティブソリッドの一つとして取扱得ることを示した。

これは、第2章で与えた曲面間の相貫曲線の式を利用することによって、CSGでは困難とされていた曲面接合部のフィレットを高精度に自動生成できることを示したものである。(第3章)

3. CSGプリミティブ間の相貫曲線を求め、その有効部分を抽出する方法を開発することによって、ソリッドモデルの二つの代表的な表現間の変換、すなわち、CSG表現からB-Reps表現への変換を解析的に行った。すなわち、

(a) 平面、円柱、円錐、球面を厳密曲面のまま扱う方法を開発した。

(b) B-Repsの形状稜線の候補として、曲面間の相貫曲線を解析的厳密解として求め、さらに、その相貫曲線と他の曲面群との間で相貫点を求めて相貫曲線を部分区間に分割することができた。

(c) 相貫曲線上の1点の有効判定式、及び同一面の存在に関係する有効性の判定式を与え、形状稜線候補の中から実際の形状稜線を取り出す方法を導いた。

(d) 形状稜線の持つ両端点の座標値、及び面に対するポインタ情報から、B-Repsのトポロジーデータを作成し、幾何データと合わせてB-Repsモデルを構築した。

さらに、上記の方法にしたがってCSGからB-Repsへの変換システムEAGLEを構築し、変換結果として得られるB-Repsモデルの構造を示した。また、いくつかの形状について変換実験を行い、その高速性と高精度性が確認された。この変換法によって、各々の特徴をいかす形のCSG/B-Repsの二重構造モデルが構築できた。(第4章)

4. プラスチック製品などの製造には欠かせない金型設計において、金型モデルを自動生成するシステムを構成するための、金型の抜取り可能判定法および分割線の自動生成法を開発した。すなわち、製品形状モデルとしてCSG/B-Reps二重構造モデルを利用し、2プレート金型を用いるという仮定のもとに、抜取り方向が与えられるものとして、その方向に抜取り可能であるかどうかを判定するために、次の(a)から(c)のことを行った。

(a) 抜取り可能の定義・定式化を行った。

(b) 定義に基づいて抜取り可能判定を行うためのアルゴリズムとして、

- 面分の分類
- 分割線分の生成
- 分割線分の可視性判定

を示した。この際、面分の分類および分割線分の生成には形状の表面情報が必要なので、B-Reps表現が有効であり、分割線分の可視性判定には高速に計算できるCSG表現が有効であった。

(c) 計算機実験によって本アルゴリズムの正当性を評価した。

(第5章)

5. 複数の部品形状あるいは組立製品を扱うためには、各部品それぞれの形状情報のみならず、各部品間の関係も認識する必要がある。すなわち、部品間にめり込みがあってはいけないし、めり込みがなくても、部品が互いにどのように接触しているかを認識できなければならない。例えば、テーブルとある物体が接触していることが分かったとしても、物体がテーブルの下面に接触しているだけならば、その物体は落下してしまうはずである。このようなことを認識できるようにする

ために、製品モデルとしてCSG/B-Reps二重構造モデルを適用して、形状モデル間の干渉認識の手法を開発した。具体的には、

(a) 3次元形状モデル間の干渉認識をするために、線分および面分の状態の分類・定式化を行った。

(b) 状態分類式を基に干渉認識を行うために形状モデルに施されるべきアルゴリズムとして、

i. 逆向き同一面の抽出（面情報が必要なためB-Reps表現を利用した。）

ii. 面分間の相貫曲線の計算（B-Reps表現から曲面分の情報を取り出し、第2章の相貫曲線式を用いた。）

iii. 多重線分の分解

iv. 線分の状態判定（CSG表現による形状内外判定法およびB-Reps表現によるトポロジー情報を利用した。）

v. 面分の状態判定（面分を囲む線分の状態およびCSG表現による形状内外判定法を用いた。）

を示した。

(c) 3次元形状モデル間の干渉認識システムを構築して実験を行い本手法の有効性を示した。

干渉認識結果を利用することにより、組立・分解作業計画の自動生成、あるいは、シミュレーションが可能となる。（第6章）

6. 第6章の干渉認識の応用として、組立製品における各部品の抜取り可能方向の決定と抜取り（組立）順序の自動生成法を開発した。この方法では、部品間の接続情報を自動的に計算するので、設計者が接続関係を定義する必要はない。また、設計者が定義したいような場合でも、間違いの自動検出などに利用できるので、自動化のより進んだシステムの構築に役立つと考えられる。（第7章）

7. CAD/CAMシステムの構築・カスタマイズ・管理をするためのCAD/CAMシステム構築環境を提案した。CAD/CAMシステムは膨大なプログラム群からなることを考えると、システム設計者がそれらすべてのプログラムを把握することは容易ではない。そこで、自律駆動型プログラムモジュール（ADPM）という概念を導入し、プログラム部品の方から積極的にシステムを構成しようとする機能をプログラム部品自身にもたせる考え方を示し、そのためのプロトタイプの試作を行った。すなわち、

(a) ADPMはオブジェクト（下位のADPMまたは既存のプログラムモジュール）、オペレーション（ADPMの働きを決めるプログラム）、および、ワーキングスペース（環境）よりなるものとして定義した。

(b) オペレーションを入力部、本体、出力部の3つに機能分解した。

(c) 初期状態、入力待状態と出力待状態を取り入れることにより、オペレーションの3つの部分が互いに連携して処理が進むように制御する方式を示した。

(d) Lisp言語による自律駆動型プログラムモジュールの構造を示した。

(e) 自律駆動型プログラムモジュールを実行させるインタープリタの制御方式を示した。

（第8章）

謝辞

本論文は、北海道大学工学部精密工学教室および情報工学教室において約5年間、さらに、京都大学工学部応用システム科学教室において約3年間にわたって行ってきた研究成果をまとめたものであります。この間、北海道大学在学中から京都大学に移動して現在に至るまでの全期間において、本研究に対して多くの御示唆を頂き、また日頃から叱咤激励くださり、さらに本論文をまとめるに当たって御懇切な御指導を頂きました京都大学工学部応用システム科学教室沖野教郎教授に心から感謝致します。

また、北海道大学在学中には、機に応じて適切なアドバイスを頂き、良き相談相手となってくださり、京都大学移動後も遠方にもかかわらず親切な御指導をして頂きました北海道大学工学部精密工学教室嘉数侑昇教授に深く感謝の意を表します。

本研究の重要な部分であるフィレット発生問題に関して多くの御助言を頂きました東海大学工学部城間祥之講師に御礼申し上げます。

最後に、本論文に関して多くの有益な御意見を頂き、また、いろいろな面でお世話になりました山本裕助教授をはじめとする京都大学工学部応用システム科学教室応用人工知能論講座の皆様に深く感謝致します。